Untangling the Heart

Automated Fiber Segmentation and Structural Metrics via Deep Learning

IN5000

Arjun Vilakathara

Delft University of Technology

TU Delft

Untangling the Heart

Automated Fiber Segmentation and Structural Metrics via Deep Learning

by

Arjun Vilakathara

To obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday October 14, 2025 at 13:45.

Instructor:	Dr. N. Tömen
Teaching Assistant:	A.C. Garcia
Student number:	4995074
Project Duration:	December 2024 - October 2025
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science, Delft

Thesis committee	Dr. N. Tömen	TU Delft, Chair
	Dr. Michael Weinmann	TU Delft, Core Member

Cover: EHT image provided by Dr. Milica Dostanic
Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Preface

This thesis is the result of my Master's research at Delft University of Technology, carried out in the Pattern Recognition and Bioinformatics group. It combines my interest in computer vision with biomedical applications, focusing on engineered heart tissues (EHTs).

Working on this project brought together ideas from multiple fields: tissue engineering, confocal microscopy, signal processing, and deep learning. My goal was not only to address the scientific challenges, but also to build a clear and reproducible pipeline in which each preprocessing step, model choice, and evaluation method is documented. I believe this transparency will make it easier for others to reproduce, extend, and apply the work.

I am deeply grateful to my supervisors Dr. Nergis Tömen, Alejandro Castañeda, Sara Cardona, and Dr. Milica Dostanic for their guidance and help through the challenges I faced during this thesis. I am especially thankful to Dr. Nergis Tömen and Alejandro Castañeda, who served as my daily supervisors and played a key role in helping me find the right direction for this thesis when it was most difficult. Their continued support, feedback, and encouragement were instrumental in shaping both the research and my growth as a researcher. I would also like to thank the thesis committee member Dr. Michael Weinmann for the interest he has shown in my research and for his evaluation of my work.

Finally, I would like to thank my family for their constant support, patience, and encouragement throughout my studies. Their belief in me made this work possible.

Arjun Vilakathara
Delft, October 2025

Summary

Engineered heart tissues (EHTs) provide a promising platform for studying cardiac physiology, but their dense fibrous architecture poses significant challenges for quantitative analysis. This thesis develops an automated pipeline for segmenting and analyzing fibers in confocal microscopy images of EHTs, with the goal of extracting structural metrics relevant to tissue function.

The pipeline integrates:

1. **Preprocessing** using FFT-based bandpass filtering to isolate mid-frequency structures corresponding to fibers.
2. **Segmentation** with three encoder–decoder architectures: U-Net, Attention U-Net, and U-Net++ (with and without pretrained backbones).
3. **Refinement** through a secondary U-Net aimed at improving continuity and suppressing false fibers.
4. **Evaluation** at both the pixel level (Dice, MSE, confusion matrix components) and fiber level (length, connectivity, orientation), complemented by human inspection.

Experiments were conducted on two datasets: a labeled synthetic collagen dataset for accurate benchmarking, and real EHT confocal volumes with sparse manual annotations. The results reveal clear trade-offs between models. U-Net recovered the most complete and connected fibers but introduced hallucinations. Attention U-Net generated clean outputs with fewer hallucinations but more fragmented fibers. U-Net++ balanced sensitivity and structural fidelity, with pretrained backbones offering the best quantitative scores. Refinement networks reduced hallucinations in U-Net outputs but often further fragmented the fibers in other models.

Fiber-level metrics and visual inspection confirmed that orientation is consistently captured, while continuity and connectivity remain major challenges. The work highlights the importance of frequency-domain preprocessing, careful model selection, and the need for improved annotation and refinement strategies.

Overall, this thesis demonstrates the feasibility of automated structural analysis of EHTs and provides a reproducible framework that bridges confocal imaging with computational modeling. The findings establish a foundation for future extensions to 3D datasets, more advanced refiners, and broader biomedical applications.

Contents

Part I

Preliminary Materials and Background

1
Engineered Heart Tissues: Background and Significance

1.1. What are Engineered Heart Tissues (EHTs)?

Engineered heart tissues (EHTs) are lab-grown constructs designed to replicate the structure and function of native cardiac muscle. They are typically developed by culturing cardiomyocytes, often derived from human induced pluripotent stem cells (hiPSCs), within a three-dimensional biomaterial scaffold [1]. Over time, these cells mature and align, leading to tissue constructs that exhibit spontaneous and paced contractile activity similar to native myocardium.

EHTs provide an in-vitro system where cellular, structural, and mechanical properties can be studied under controlled laboratory conditions. Unlike two-dimensional cell cultures, they preserve a physiologically relevant three-dimensional environment, allowing realistic investigation of cardiac physiology and pathology [2, 3].

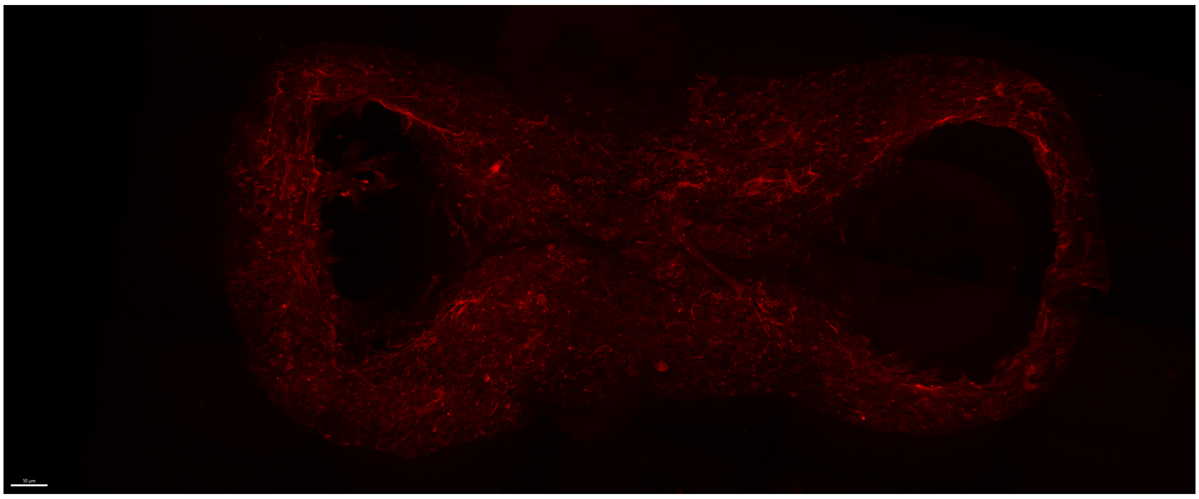


Figure 1.1: An image of a EHT tissue captured using confocal microscopy.

1.2. Why Fiber Architecture Matters

A defining feature of cardiac tissue is its fibrous architecture. Fiber orientation and connectivity govern the anisotropic propagation of electrical signals and the mechanical response under load [4, 5]. In EHTs, understanding the internal fiber organization is critical for linking microstructure to macroscopic contractile performance [3].

Metrics such as fiber length, orientation, and connectivity can thus be used to estimate and evaluate mechanical robustness. Highly aligned fibers tend to produce stronger, more coordinated contractions, whereas disordered or fragmented fibers reduce contractile efficiency. For computational modeling and biomechanical simulations, accurate quantification of these structural properties is essential.

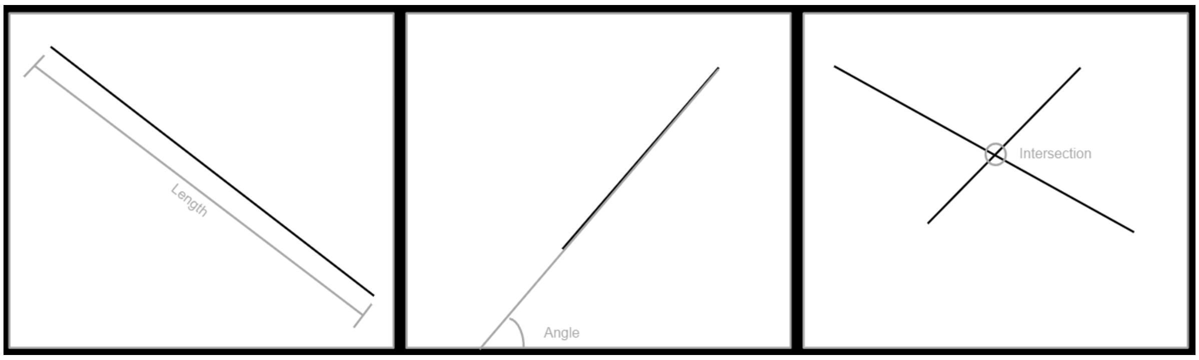


Figure 1.2: Example representation of fibrous tissue with structural metrics. Length is measured as the span of individual fibers, angle quantifies fiber orientation relative to an axis, and connectivity denotes intersections where fibers meet. Such metrics capture biologically meaningful features of EHT organization.

1.3. Use Cases and Applications

EHTs are useful and critical in several domains.

- Drug screening: EHTs provide a human-relevant platform for testing cardiotoxicity and drug efficacy without relying on animal models.
- Disease modeling: Patient-derived hiPSCs enable EHTs that replicate genetic cardiac diseases in-vitro, making them a valuable research tool.
- Regenerative medicine: Insights from EHTs provide insight into replacing damaged cardiac tissue.

Their reproducibility and controlled culture conditions make EHTs particularly attractive for computational integration. Imaging techniques such as confocal microscopy provide detailed imagery of tissue architecture, which can be fed into automated pipelines for segmentation and analysis.

Summary

Their dense fibrous architecture of EHT's are both a biological and a computational challenge. The remainder of this thesis develops and evaluates automated approaches to extract, segment, and analyse this structure, enabling further research into their mechanical properties.

2 Imaging and Computer Vision

2.1. Confocal Microscopy

Confocal microscopy is a laser-scanning technique that provides optical sectioning of thick biological samples by rejecting out-of-focus light through the use of a pinhole aperture. Compared to conventional widefield microscopy, confocal imaging yields higher contrast and resolution in thick specimens, making it particularly suited for three-dimensional tissue constructs such as engineered heart tissues (EHTs). By acquiring serial optical sections, confocal microscopy can generate volumetric image stacks that capture both cellular and extracellular features with sub-micron precision [6].

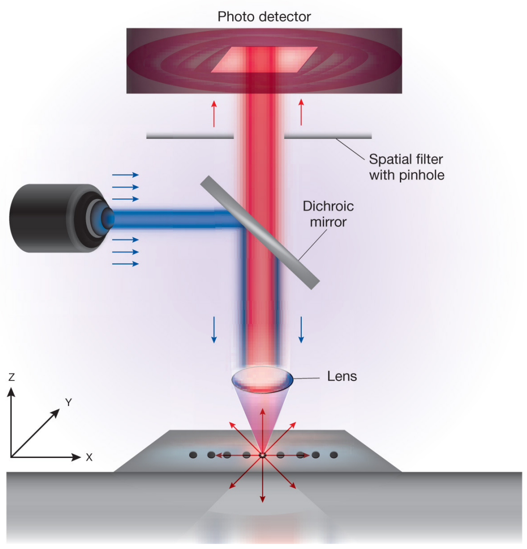


Figure 2.1: Schematic of a confocal microscope. A laser beam is focused into the sample, and emitted light from the focal plane is collected through a pinhole before detection. (Image from [6]).

2.2. Computer Vision Context

From a computer vision perspective, confocal stacks can be treated as dense grayscale images with high dynamic range and significant background noise. They exhibit:

- **High resolution and detail**
- **Low signal-to-noise ratio** in certain regions
- **High structural complexity** due to overlapping fibers and heterogeneous textures.

These properties make automated analysis challenging: segmentation models must separate fine fibrous structures from noisy backgrounds, maintain continuity across slices, and preserve geometrical features relevant to downstream biomechanical modeling.

2.3. Image Processing

Before applying deep learning methods, the images/slices retrieved from the microscope must first be processed to remove unwanted information or to make relevant information stand out more.

2.3.1. Noise and Normalization

Raw confocal microscopy images often contain a nearly constant background intensity caused by autofluorescence, detector offsets, or scattered light. This background makes it harder to distinguish true structures (fibers) from noise. A simple way to correct for this is background subtraction.

let $I(x, y)$ be the raw grayscale intensity at pixel coordinates (x, y). We estimate a background value b by sampling regions of the image that contain no visible structures (e.g., the corners or dark areas). The corrected image is then

$$I_{\textsf{sub}}(x, y) = I(x, y) - b.$$

If subtraction leads to negative values, these are clipped to zero:

$$I_{\textsf{sub}}(x, y) = \max\{I(x, y) - b,\ 0\}.$$

After background subtraction, pixel values may have arbitrary units or ranges depending on the microscope's settings. To make data comparable across images and suitable for computer vision models, we apply normalization. The simplest form, which we use, is min–max normalization:

$$I_{\textsf{norm}}(x, y) = \frac{I_{\textsf{sub}}(x, y) - I_{\min}}{I_{\max} - I_{\min}} \cdot 255,$$

where $I_{\min}$ and $I_{\max}$ are the minimum and maximum pixel intensities in the background-subtracted image. This rescales the data into the standard range $[0, 255]$ used for 8-bit grayscale images.

Normalization has two advantages:

- It makes brightness values comparable across different images, regardless of acquisition settings.
- It ensures consistency of input images when training deep learning models, which typically expect inputs in a fixed range (e.g., $[0, 1]$ or $[0, 255]$).

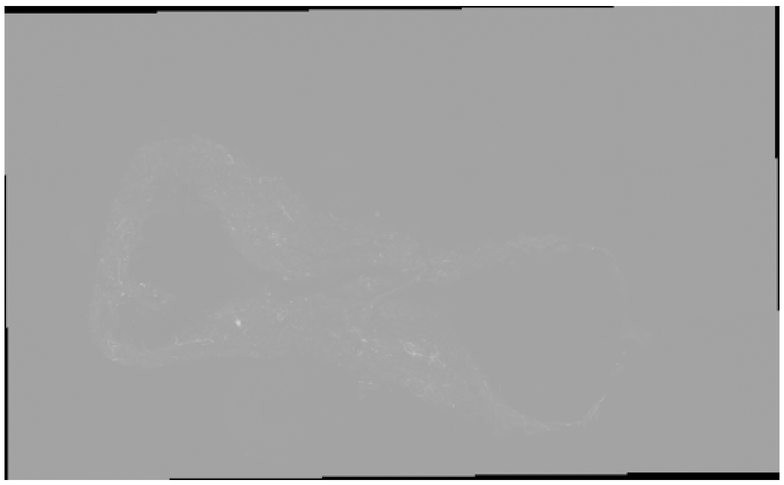


Figure 2.2: Example of raw output of microscope with background noise.

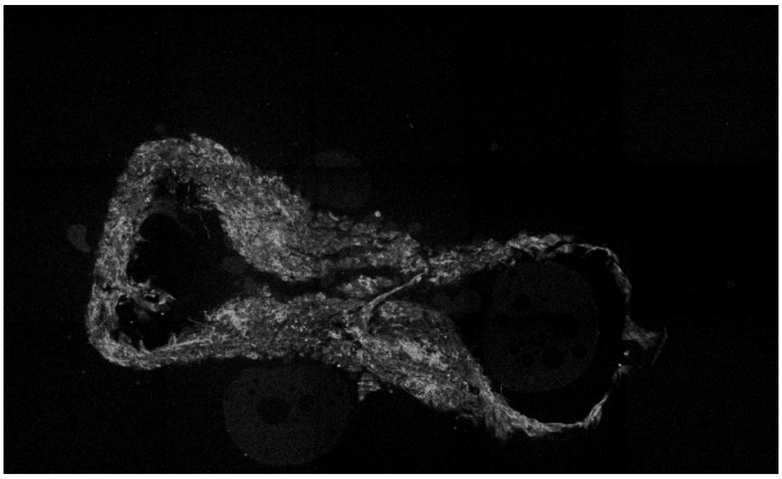


Figure 2.3: Example of raw output in image 2.2 with the background subtracted and then normalized

2.3.2. Frequency Content, FFT

Every grayscale image can be understood not only as a grid of pixel intensities but also as a sum of spatial frequency components:

- **Low frequencies** correspond to slow intensity changes across the image, such as the smooth gradient of the sky in a portrait photo.
- **High frequencies** correspond to very rapid changes, such as noise or sharp edges, like the fine details of leaves on a tree.
- **Mid frequencies** often capture structural features like fibers: thin but extended patterns that vary neither too slowly nor too rapidly.

Mathematically, this decomposition is achieved through the two-dimensional Discrete Fourier Transform (DFT), which represents the image in terms of its sinusoidal frequency components.

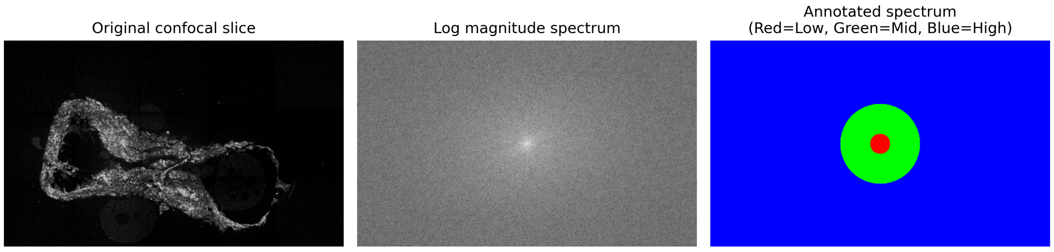


Figure 2.4: Left: original confocal image slice $f(x, y)$. Middle: log-scaled magnitude spectrum $|F(u, v)|$, showing frequency distribution. Right: annotated spectrum highlighting low, mid, and high frequency regions.

The Fourier Transform of an Image. Let $f(x, y)$ be an image of size $M \times N$. The two-dimensional Discrete Fourier Transform (DFT) is defined as:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-2\pi i\left(\frac{ux}{M} + \frac{vy}{N}\right)},$$

where (u, v) are frequency coordinates. $F(u, v)$ is generally complex-valued:

$$F(u,v) = A(u,v)\, e^{i\phi(u,v)},$$

where $A(u, v) = |F(u, v)|$ is the **magnitude spectrum** and $\phi(u, v)$ is the **phase spectrum**.

Magnitude Spectrum. The magnitude $A(u, v)$ tells us how much of each frequency is present in the image.

- Bright spots near the center of the spectrum correspond to low frequencies (large-scale intensity variations).
- Features away from the center correspond to higher frequencies.

Filtering in the Frequency Domain. We can isolate specific frequency ranges by multiplying the Fourier spectrum with a mask $H(u, v)$:

$$\widetilde{F}(u, v) = F(u, v) \cdot H(u, v).$$

Typical masks include:

- **Low-pass:** $H(u, v) = 1$ for small $\sqrt{u^2 + v^2}$, else 0 (keeps only low frequencies).
- **High-pass:** $H(u, v) = 1$ for large $\sqrt{u^2 + v^2}$, else 0 (keeps only fine details).
- **Band-pass:** $H(u, v) = 1$ only within a range between radii R_{low} and R_{high}.

Figure 2.5 shows what each filter looks like on the magnitude spectrum from figure 2.4. White is where the filter is 1 and black is where it is 0.

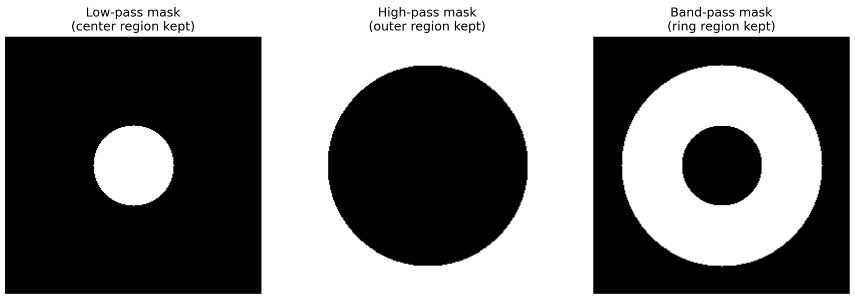
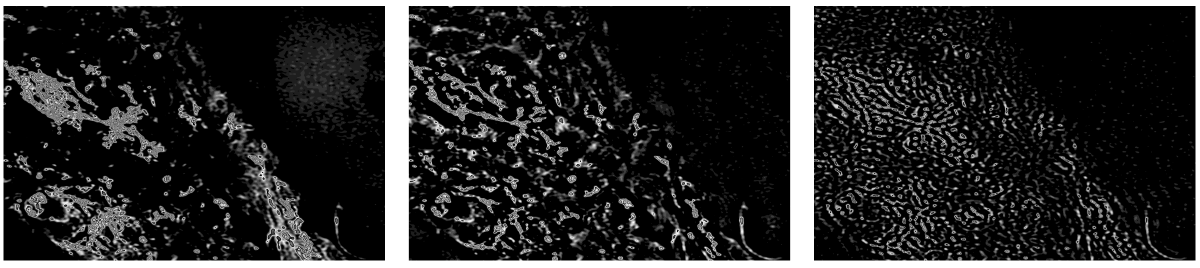


Figure 2.5: Examples of frequency masks. Left: low-pass (center region kept). Middle: high-pass (outer region kept). Right: band-pass (ring-shaped region kept).

Inverse Transform. After masking, we convert back to the spatial domain using the inverse Fourier transform:

$$\tilde{f}(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \widetilde{F}(u, v)\, e^{+2\pi i \left(\frac{ux}{M} + \frac{vy}{N}\right)}.$$

The result $\tilde{f}(x, y)$ is a filtered image that contains only the desired frequency content. For fiber segmentation, we choose band-pass filters that preserve mid-frequency structures corresponding to fibers, while discarding low-frequency background and high-frequency pixel noise.



(a) Low-cut = 0 **(b)** Low-cut = 60 **(c)** Low-cut = 200

Figure 2.6: Frequency filtering example with fixed high-cut value. As the low-cut threshold increases, more low-frequency background is removed, progressively isolating mid-frequency fiber content.

Figure 2.6 illustrates the effect of applying FFT-based filtering with a fixed high-cut frequency while gradually increasing the low-cut threshold. As the low-cut increases, more of the lower frequencies are removed. Initially this helps suppress background intensity variations and illumination gradients. However, when the low-cut is pushed too high, we begin to lose mid-to-high frequency content that actually corresponds to the fibrous structures of interest. At very high low-cut values (e.g., 200), the fibers are almost completely lost, leaving primarily residual noise. This demonstrates that while low-cut filtering can be beneficial, retaining a sufficient range of high-frequency information is essential to preserve fiber visibility and structural integrity.

2.4. Segmentation in Computer Vision

When addressing the general task of defining what's in an image, we can define it as assigning a label to an entire image (e.g., classifying whether it contains a cat or a dog). Image segmentation, in contrast, is defined as the process of assigning a label to every individual pixel in the image. Formally, given an input image $I \in \mathbb{R}^{H\times W}$, where H and W are the height and width of the image respectively, the goal is to learn a function

$$f : \mathbb{R}^{H\times W} \longrightarrow \{0,1\}^{H\times W},$$

which maps each pixel location (x, y) to either foreground (feature of choice such as a cat) or background. The output of such a model is therefore a binary mask of the same resolution as the input, where each pixel encodes the presence or absence of the structure of interest. In multi-class segmentation, f can map to more than two categories, but in our case of engineered heart tissue (EHT) images we restrict to a binary setting. Figure 2.7 illustrates this concept, showing how the input image is decomposed into regions of interest by a segmentation model.

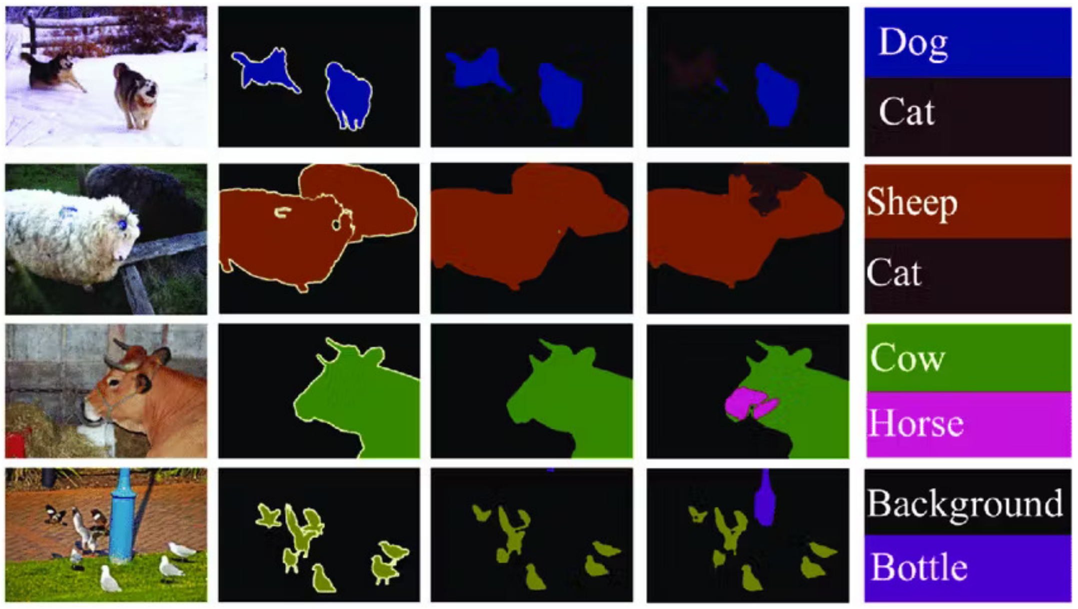
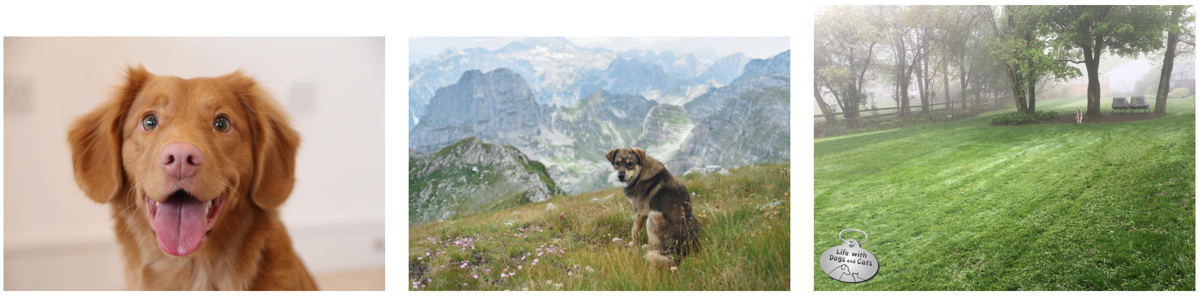


Figure 2.7: Example of multi-class segmentation. The input photograph (left) is paired with a pixel-wise segmentation mask (right), where each colour encodes a distinct semantic class (e.g., cow, horse, person, vegetation, sky, ground), and background is treated as its own class. While this figure illustrates the general task of semantic segmentation, this thesis focuses on the problem of fiber vs. background using an identical input–output structure.

Challenges in biomedical and microscopy segmentation. Segmentation in natural images, such as a picture of a dog, primarily requires preserving the fine boundaries between the foreground object and its background. Biomedical microscopy, however, presents additional challenges. Confocal slices are typically grayscale, lacking the rich color information available in natural image segmentation. Moreover, the signal-to-noise ratio (SNR) is often low. In practical terms, this ratio describes the ratio of pixels belonging to the structure of interest versus the background. Figure 2.8 illustrates this concept with three images of a dog at different SNR levels: as the ratio decreases, even human observers find it increasingly difficult to separate signal from noise.



(a) High SNR: dog clearly visible in foreground. Image from [7]

(b) Medium SNR: dog visible but further in background. Image from [8]

(c) Low SNR: dog is very hard to distinguish from from background. Image from [9]

Figure 2.8: Illustration of signal-to-noise ratio (SNR) using a natural image example. As SNR decreases, the object of interest (the dog) becomes progressively harder to distinguish from background noise.

For medical images, this problem is often much more severe. In some cases, the ratio may be as low as 1,000 foreground pixels for 20,000 background pixels. On top of this imbalance, detector noise, autofluorescence, and light scattering further obscure faint structures. In such settings, traditional approaches such as edge detection or morphological filtering are insufficient. Instead, deep learning models must learn the weak but consistent patterns while suppressing the abundant background noise.

This leads into the other challenge of class imbalance. In the fibrous tissues used in this thesis, fibers often constitute less than 10% of all pixels, With the overwhelming majority belonging to background. This imbalance can bias models toward predictions where nearly every pixel is classified as background, yielding deceptively high accuracy but biologically meaningless results. Viewed from the perspective of signal-to-noise ratio (SNR), the signal i.e., the pixels corresponding to fibers, is severely diluted by the noise of background pixels. For example, a ratio of 1,000 foreground pixels to 20,000 background pixels represents an SNR so low that distinguishing structure from noise is difficult even for human observers. In such cases, models must be robust to the scarcity of positive pixels and the abundance of background/negetive pixels.

Specific challenges in fiber segmentation. Fiber segmentation introduces unique difficulties beyond generic biomedical imaging. Fibers are thin, often only a few pixels wide, and their continuity across long distances is crucial for downstream biomechanical analysis. For example, metrics such as fiber length and connectivity depend on recovering as much of a connected fiber as possible, in contrast to fragments of it. False positives, or “hallucinated fibers,” are especially problematic in this context: they may inflate connectivity metrics or introduce non-existent junctions, leading to misleading conclusions about tissue structure.

Summary

Confocal microscopy provides high-resolution volumetric images of engineered heart tissues but introduces challenges such as noise, intensity variations, and dense structural complexity. Traditional image processing techniques (edges, morphology, frequency filtering) offer useful insights but are insufficient on their own. This motivates the application of deep learning models, which perform the task of segmentation, as introduced in the subsequent chapters.

3

Deep Learning and CNN Basics

3.1. Nonlinear Activation Functions

Neural networks rely on nonlinear activation functions. Without nonlinearity, a sequence of linear layers would collapse into a single linear transformation, severely limiting the representational power of the model. Activation functions introduce nonlinear decision boundaries, enabling the network to approximate more complex mappings from inputs to outputs.

Commonly used functions include:

- **Sigmoid:** squashes inputs to $(0, 1)$, historically popular but prone to saturation and vanishing gradients.
- **Tanh:** similar to sigmoid but symmetric about zero, still affected by gradient saturation.
- **ReLU (Rectified Linear Unit):** $f(x) = \max(0, x)$, avoids saturation for positive values and is computationally efficient.

ReLU is the default choice in most modern CNNs, as it supports sparse activations and accelerates convergence [10].

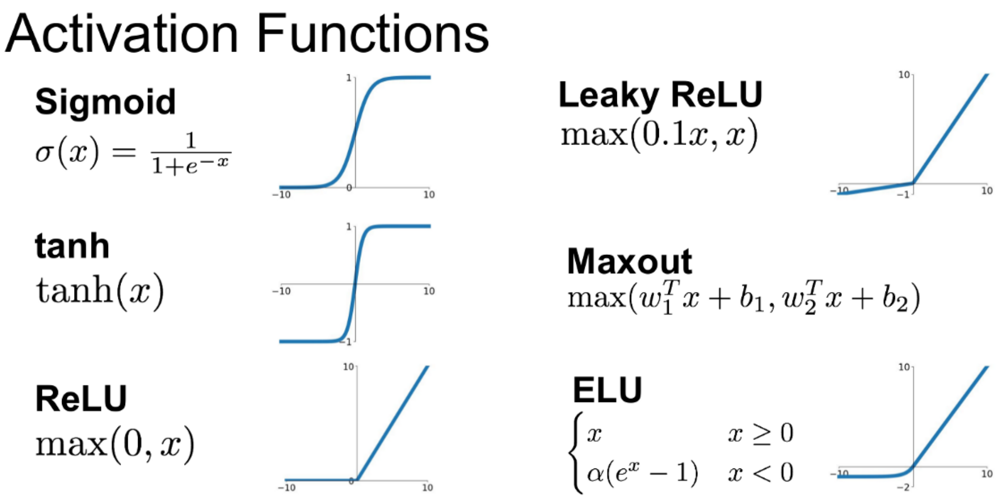


Figure 3.1: Examples of nonlinear activation functions. Image from [11].

3.2. From Perceptrons to Deep Networks

Neural networks are composed of computational units called neurons. A single neuron, also known as a perceptron, maps an input vector $\mathbf{x} \in \mathbb{R}^d$ to a scalar by applying a weighted sum with bias, followed by a nonlinear activation:

$$z = \mathbf{w}^\top \mathbf{x} + b, \qquad a = \sigma(z),$$

where $\mathbf{w} \in \mathbb{R}^d$ is the weight vector, $b \in \mathbb{R}$ is the bias, and $\sigma(\cdot)$ is an activation function such as ReLU or sigmoid.

From Perceptrons to Deep Networks. While a single perceptron outputs only one scalar, stacking multiple perceptrons in parallel forms a layer, producing a vector of activations. A fully-connected layer with m neurons maps

$$\mathbf{x} \in \mathbb{R}^d \quad \longmapsto \quad \mathbf{a} = \sigma(W\mathbf{x} + \mathbf{b}),$$

where $W \in \mathbb{R}^{m \times d}$ is a weight matrix, $\mathbf{b} \in \mathbb{R}^m$ is a bias vector, and σ is applied element-wise.

Deep networks are built by connecting such layers together:

$$f(\mathbf{x}) = f^{(L)}\big(f^{(L-1)}(\cdots f^{(1)}(\mathbf{x}))\big),$$

where each $f^{(\ell)}$ is a nonlinear transformation parameterized by weights and biases. Stacking many such layers, a process referred to as increasing the depth of the network, improves its representational capacity and enables it to approximate highly complex functions. Naturally, adjusting the depth of a network comes with trade-offs: increasing depth allows the model to capture more abstract and complex patterns, but also raises the risk of overfitting, vanishing gradients, and higher computational cost, whereas shallower networks are easier to train but may lack in capacity for more complex tasks.

Limitations of Fully Connected Layers. A fully connected layer is one in which every neuron in a given layer is connected to every neuron in the preceding and succeeding layers. Each such connection is associated with a weight, and each neuron typically also has an additional bias term. The total number of trainable parameters in a layer is therefore determined by the number of connections (weights) plus the number of biases. This means that parameter count scales as the product of the input and output sizes of the layer.

This is why densely connected layers, even if not fully connected, become impractical when applied directly to images, including relatively small ones. The parameter count still scales with the product of input and output dimensions, and in the case of segmentation, where each pixel is both an input feature and must also be predicted as an output, this problem is highlighted further.

As an example, for an input image of size $H \times W$, the input dimension is $d = H \cdot W$. Mapping this input to a hidden layer of m neurons requires $d \times m$ weights, in addition to m biases. A 256×256 grayscale image has 65,536 pixels. Even mapping to a hidden layer of only 1,000 neurons would require $65{,}536 \times 1{,}000 \approx 65$ million weights, plus 1,000 biases, in a single layer. Such a dense parameterization not only demands excessive memory and computational power, but also introduces several practical and theoretical problems:

- **Overfitting:** With so many connections, the network can memorize the training images instead of learning general patterns. In practice, this means it performs very well on the training data but fails to generalize to new, unseen examples.
- **Vanishing gradients:** In very deep networks, the error signals used for learning gradually diminish as they are propagated backward through the layers, a process known as backpropagation (explained in Section 3.5.3). In simple terms, the earliest layers receive almost no useful feedback, causing them to stop learning effectively.
- **Computational cost:** Tens of millions of connections require large amounts of memory and processing. This makes training extremely slow and inefficient as image sizes grow.

Convolution as a Solution.

Convolution as a Solution. Both fully connected layers and convolutional layers perform the same kind of operation: they take an input, apply a linear transformation with trainable weights and biases, and then pass the result through a nonlinearity. The difference then lies in the structure of the weight matrix. In a dense layer, every input pixel is connected to every output unit, leading to a weight matrix of size $(m \times d)$ where d is the number of input pixels and m the number of output units. By contrast, a convolutional layer replaces this dense weight matrix with a much smaller 'kernel' (e.g., $3{\times}3$) that is applied repeatedly across the image. Unlike fully connected layers, a convolution keeps the input and output in the same grid-like shape: it maps an image to another image-shaped feature map, instead of collapsing everything into a flat vector.

These convolutional layers address the limitations of the dense layers by introducing structure into how 'connections' are formed. Instead of every neuron being connected to every pixel, convolutions exploit two key principles:

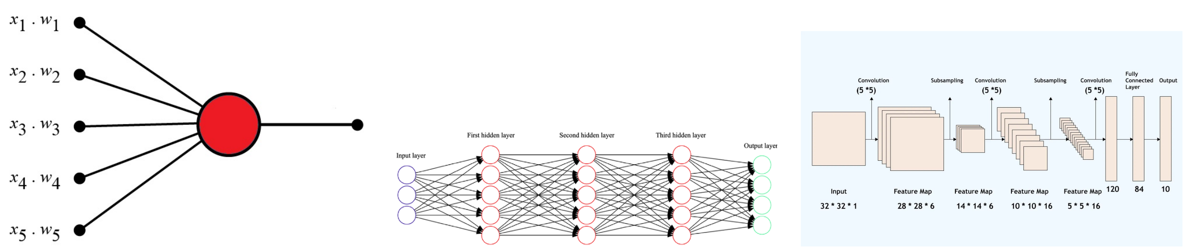
- **Local receptive fields:** each neuron connects only to a small neighborhood of the image (e.g., a $3{\times}3$ patch). This drastically reduces the number of parameters while reflecting the fact that pixels close to one another are more likely to be related than distant ones.
- **Weight sharing:** the same kernel (set of weights) is applied across all spatial locations i.e, all pixels. This ensures translation equivariance: For example, a filter that learns to detect the outline of a dog's nose in one part of the image will respond to the same feature even if the dog appears in a different location.

This architecture directly addresses the problems of fully connected layers with images. By reducing parameters through locality and weight sharing, convolutional networks are less prone to overfitting: they cannot simply memorize the training data pixel-by-pixel, but must instead learn general patterns (e.g., edges, textures) that are useful across the whole image. Because the kernels (trained weights) are reused across locations, the model generalizes better to new inputs where the same structures may appear in different positions. The reduced parameter count also lowers computational cost, making training feasible even for large images. Finally, the hierarchical stacking of convolutional layers helps reduce the vanishing gradient problem: earlier layers focus on simple features like edges, while deeper layers combine them into complex shapes and objects, allowing gradients to propagate meaningfully through multiple layers and types of representation.

Formally, given an input feature map X and a kernel $K \in \mathbb{R}^{k \times k}$, the convolution output at pixel (i, j) is

$$Y(i,j) = \sum_{u=0}^{k-1} \sum_{v=0}^{k-1} K(u,v)\, X(i+u-p,\, j+v-q),$$

where k is a chosen kernel size (for example, $k = 3$ for a $3{\times}3$ filter), and (p, q) denotes the padding applied at the image boundaries. Padding typically means adding artificial pixels around the border, commonly zeros but sometimes mirrored values, so that the convolution can still be computed at the edges of the image. Subtracting p and q in the indexing simply shifts the kernel back into the valid image region, ensuring that every output pixel corresponds to a well-defined neighborhood.



(a) Single perceptron from [12] **(b)** Fully-connected network from [13] **(c)** Convolutional network from [14]

Figure 3.2: From perceptron to deep learning for images. Left: a single neuron (perceptron). Middle: stacking many perceptrons yields fully-connected deep networks, but with dense parameterization. Right: convolutional networks exploit local receptive fields and weight sharing to efficiently handle image data.

3.3. Convolution for Images: From Toy Example to General Form

Convolution in Practice. To illustrate convolution intuitively, consider a 5×5 image X and a 3×3 kernel K. At each spatial location, the kernel overlays a patch of the image and we compute the sum of element-wise multiplications. For example, if the kernel is

$$K = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix},$$

then sliding it over a 3×3 region of X produces

$$Y(i,j) = \sum_{u=0}^{2}\sum_{v=0}^{2} K(u,v)\, X(i+u, j+v).$$

This operation extracts vertical edge patterns: positive weights on the left subtract negative weights on the right. Repeating this across all positions forms a new feature map Y that highlights edges in the input image.

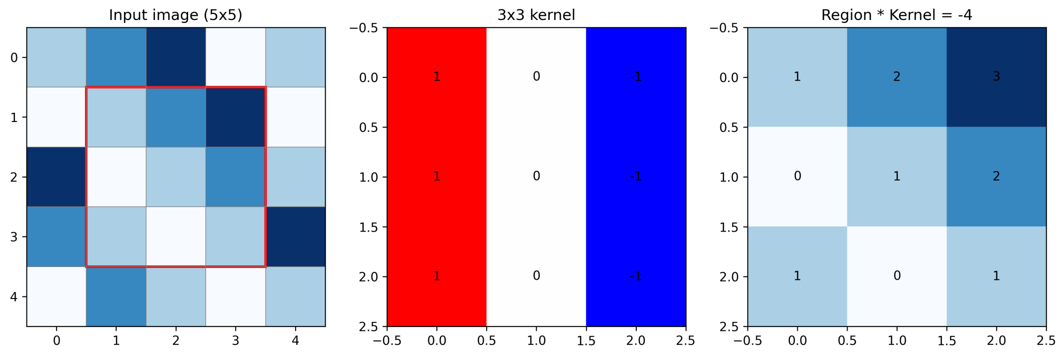


Figure 3.3: Illustration of convolution. Left: input image with a 3×3 region highlighted in red. Middle: kernel values (edge detector). Right: overlay of the kernel on the region and the computed convolution result. This shows how convolution extracts local features by combining input pixels with shared weights.

The figure above makes two points clear: (i) convolutions use local receptive fields, each output depends only on a small neighborhood of the input, and (ii) the same kernel weights are shared across all spatial locations, drastically reducing parameters compared to fully connected layers. In real networks, dozens of kernels are learned in parallel, producing multiple feature maps that capture edges, textures, and higher-level patterns. Stacking layers of convolutions allows the receptive field to grow, enabling the network to combine local features into global structures.

General 2D Convolution for Images. Fully connected layers ignore the spatial structure of images, while convolutions explicitly exploit it through locality and weight sharing. Given input image (or feature map) $X \in \mathbb{R}^{H\times W}$ and a learnable kernel $K \in \mathbb{R}^{k\times k}$, a 2D discrete convolution producing output $Y \in \mathbb{R}^{H'\times W'}$ is

$$Y(i,j) \;=\; \sum_{u=0}^{k-1}\sum_{v=0}^{k-1} K(u,v)\, X(i+u-p,\; j+v-q),$$

where (p,q) are padding terms. For multi-channel inputs $X \in \mathbb{R}^{H\times W\times C_{\text{in}}}$ and C_{out} kernels $K^{(c)} \in \mathbb{R}^{k\times k\times C_{\text{in}}}$, the output is $Y \in \mathbb{R}^{H'\times W'\times C_{\text{out}}}$ with

$$Y_c(i,j) = \sum_{m=1}^{C_{\text{in}}}\sum_{u,v} K_m^{(c)}(u,v)\, X_m(i+u-p,\; j+v-q) \quad \text{for } c = 1,\ldots,C_{\text{out}}.$$

Padding, Stride, and Receptive Field. Convolutional layers have a few key hyperparameters, which determine how the filter moves across the image and how much of the image each output unit can “see”.

- **Padding (p):** determines what happens at the image borders. Without padding, the filter cannot slide over the edges, so the output image shrinks after every convolution. Padding adds extra pixels around the border, commonly zeros, but sometimes mirrored values, so that edge pixels are treated the same as those in the center. With such paddings, the height and width of the output remain equal to the input ($H' = H$, $W' = W$).
- **Stride (s):** specifies how far the filter moves at each step. A stride of 1 means the filter slides one pixel at a time, producing a dense output. Larger strides skip pixels and therefore downsample the image, reducing its resolution. Mathematically, the output height is

$$H' = \left\lfloor \frac{H + 2p - k}{s} \right\rfloor + 1,$$

  with a similar expression for the width W'. In simple terms: larger strides mean fewer output pixels and more aggressive compression of spatial information.
- **Pooling:** another common way to reduce spatial resolution is pooling. Instead of sliding a filter with stride, pooling layers explicitly combine values within a small window (e.g., 2×2). The most common forms are max pooling, which takes the maximum value in the window, and average pooling, which takes the mean of the window. Pooling discards exact pixel values while retaining the strongest or most representative signal. In modern architectures, downsampling is often achieved by either using stride in convolutions or by applying pooling operations. They both serve the purpose of reducing resolution while increasing the receptive field.
- **Receptive field:** the portion of the input image that affects a single output unit. At shallow layers this region is small, covering only local neighborhoods (e.g., a 3×3 patch). As more layers are stacked, receptive fields expand: a unit in a deeper layer may depend on pixels spanning a much larger area of the original input. This allows the network to gradually combine fine details (edges, textures) into higher-level concepts (shapes, objects), giving it global understanding from local features.

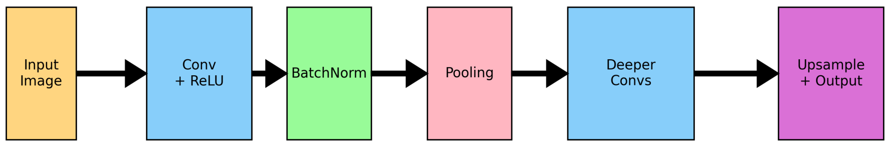


Figure 3.4: Basic CNN pipeline: convolution (+ReLU), optional batch normalization [15], and downsampling (stride or pooling), followed by deeper blocks. For segmentation, we keep spatial grids all the way to a dense per-pixel prediction.

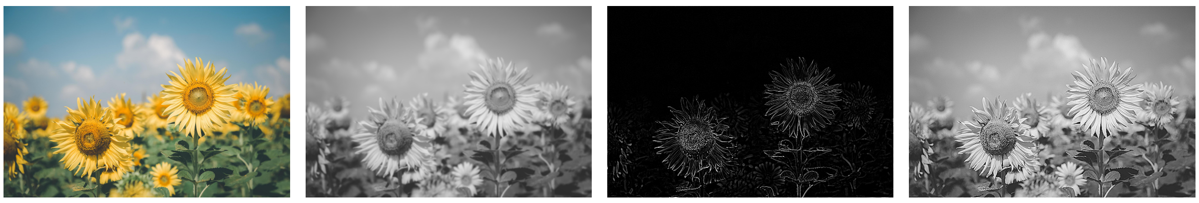
The CNN Pipeline in Words. Figure 3.4 shows how the core components of a convolutional neural network work together. An input image is first processed by a number of convolutional filters, each detecting a different type of local feature such as edges, corners, or textures. After each convolution, a nonlinear activation function (typically ReLU) is applied, introducing nonlinearity so that the network can model complex decision boundaries. Batch normalization layers are often included to stabilize the distribution of activations, making optimization faster and more reliable. Downsampling, either through strided convolutions or pooling, progressively reduces the spatial resolution while expanding the number of feature channels. This allows the network to aggregate local features into increasingly abstract, global representations. Stacking multiple such blocks deepens the receptive field: shallow layers capture simple edges, while deeper layers encode higher-level structures. For segmentation tasks, unlike classification, the spatial grid is preserved all the way to the output, so that the network can produce dense, per-pixel predictions that map directly back onto the input image domain.

3.4. Convolution as Image Filtering

Up to now, we have described convolution mathematically. To build intuition, it helps to view convolutions as filters applied to images. In classic image processing, we often use small kernels (e.g., 3×3 or 5×5) to detect edges, sharpen detail, or blur noise. In convolutional neural networks, the same principle applies, except now the filters are learned automatically through the training process rather than designed by hand.

Visual Example: Classic Filters. Figure 3.5 illustrates this idea using a natural image (sunflower field). Applying different 3×3 kernels yields distinct effects:

- The blur kernel produces a smoothed version of the field, suppressing fine texture.
- The edge kernel highlights the outlines of sunflower stems and leaves.
- The sharpen kernel enhances petal details while increasing contrast noise.



(a) Original **(b)** Blur filter **(c)** Edge filter **(d)** Sharpen filter

Figure 3.5: Convolutions as image filters. Classic kernels applied to a sunflower field produce blur, edge detection, and sharpening. In CNNs, similar filters are not designed manually but *learned automatically* from data.

Custom Filters for Targeted Patterns. Beyond standard edge or blur filters, kernels can also be designed to emphasize specific structures at different scales. For illustration, we applied two Laplacian-of-Gaussian (LoG) style kernels of different sizes to a natural scene containing sunflowers (Figure 3.6). The larger kernel responds to broad, blob-like structures such as the overall flower heads, while the smaller kernel emphasizes finer-scale detail. The comparison shows how simply changing the kernel size shifts the sensitivity of the filter to different aspects of the image. This demonstrates the principle that convolution acts as a detector of targeted patterns, with the choice of kernel directly determining what features are enhanced or suppressed.

From Hand-Crafted Filters to Learned Representations. The examples above show how different kernels emphasize different aspects of an image: blur filters suppress detail, edge filters highlight boundaries, and LoG filters emphasize blob-like structures at different scales. In traditional computer vision, experts manually selected or engineered such filters depending on the task at hand.

Convolutional neural networks (CNNs) replace this manual process by learning filters automatically from data. During training, the network adjusts the weights in its kernels so that they respond strongly to features useful for the target task, such as distinguishing fibers from background. Many first-layer CNN filters resemble familiar patterns such as edges, blobs, or Gabor filters [16], while deeper layers capture more abstract features that are not visually obvious to humans yet prove highly discriminative.

By analogy to the sunflower example, CNNs can learn filters that highlight subtle structures in biomedical images, such as faint fibers embedded in noisy confocal slices, that may be difficult for a human observer to consistently identify. These intermediate feature maps effectively act as automatic detectors, making downstream classification or segmentation easier. Repeated convolutional layers refine these feature maps into higher-level outputs, ultimately producing either segmentation masks or labels corresponding to the structures of interest.

3.5. Training by Gradient-Based Optimization

Training a convolutional neural network consists of repeatedly simulating predictions, measuring error, and adjusting parameters to reduce that error. This process can be broken down into the forward pass, loss evaluation, backpropagation, and parameter updates.

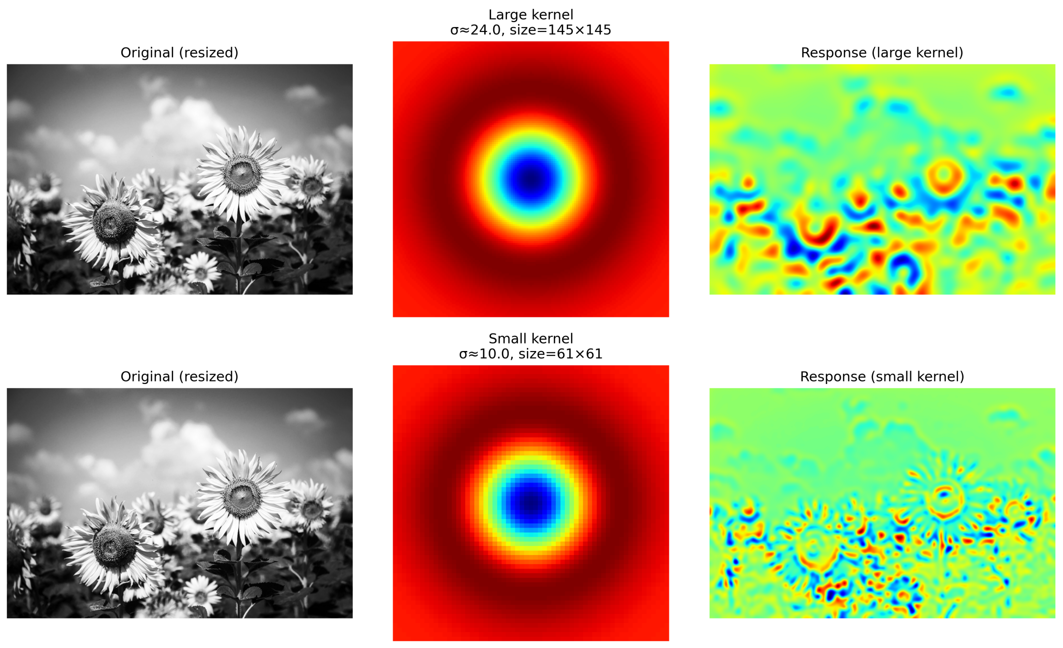


Figure 3.6: Effect of applying custom kernels to a natural image. Top row: a large kernel highlights broad, blob-like structures. Bottom row: a small kernel emphasizes fine-scale detail. This illustrates how convolutional filters can be tuned to extract different patterns from the same input image.

3.5.1. Forward Pass

Let the input image be $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$, where H and W are height and width, and C is the number of channels. Each layer of the network computes a nonlinear transformation:

$$\mathbf{h}^{(\ell)} = f^{(\ell)}(\mathbf{h}^{(\ell-1)}; \theta^{(\ell)}), \qquad \mathbf{h}^{(0)} = \mathbf{x},$$

where $\theta^{(\ell)}$ are the trainable parameters of layer ℓ. In convolutional layers, this transformation is a discrete convolution followed by a nonlinearity:

$$\mathbf{h}^{(\ell)}(i,j,c) = \sigma\left(\sum_{m=1}^{C_{\ell-1}} \sum_{u,v} K^{(\ell)}_{u,v,m,c}\, \mathbf{h}^{(\ell-1)}(i+u, j+v, m) + b_c^{(\ell)}\right),$$

where $K^{(\ell)}$ is the convolutional kernel, $b_c^{(\ell)}$ is a bias, and $\sigma(\cdot)$ is an activation function such as ReLU. The final layer produces per-pixel predictions $\hat{y}_{ij} \in [0,1]$ via a 1×1 convolution and sigmoid activation.

3.5.2. Loss Function

Given ground truth labels $y_{ij} \in \{0,1\}$, the loss function quantifies prediction error. Binary Cross-Entropy (BCE) is defined as

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{HW} \sum_{i,j} \big[y_{ij} \log \hat{y}_{ij} + (1-y_{ij}) \log(1-\hat{y}_{ij}) \big].$$

For imbalanced segmentation, the Dice coefficient is more robust:

$$\text{Dice} = \frac{2\sum_{i,j} \hat{y}_{ij} y_{ij} + \epsilon}{\sum_{i,j} \hat{y}_{ij} + \sum_{i,j} y_{ij} + \epsilon}, \quad \mathcal{L}_{\text{Dice}} = 1 - \text{Dice}.$$

We use a combined loss

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{BCE}} + \mathcal{L}_{\text{Dice}},$$

which enforces both pixel-wise fidelity and region-level overlap.

3.5.3. Backpropagation

To minimize the loss, we require gradients of $\mathcal{L}$ with respect to each parameter θ. Backpropagation applies the chain rule of calculus in reverse through the network:

$$\frac{\partial \mathcal{L}}{\partial \theta^{(\ell)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(\ell)}} \cdot \frac{\partial \mathbf{h}^{(\ell)}}{\partial \theta^{(\ell)}}.$$

Intermediate gradients are recursively propagated:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(\ell-1)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(\ell)}} \cdot \frac{\partial \mathbf{h}^{(\ell)}}{\partial \mathbf{h}^{(\ell-1)}}.$$

This efficiently computes derivatives for all layers in time linear to the depth of the network.

3.5.4. Parameter Updates with Adam

Parameters are updated iteratively using an optimizer. We employ Adam [17], which adaptively scales updates for each parameter using running averages of gradients and squared gradients:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2,$$

where $g_t = \nabla_\theta \mathcal{L}_t$ is the gradient at iteration t. Bias-corrected estimates are

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t},$$

and the parameter update is

$$\theta_t = \theta_{t-1} - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}.$$

Adam combines the benefits of momentum (smoothing updates over time) and adaptive learning rates (scaling updates per parameter), which leads to faster and more stable convergence than plain stochastic gradient descent.

3.5.5. One Training Step, Visually: From Loss to Update

To visualize how the above steps work, consider a single training iteration with a mini-batch $\mathcal{B} = \{(x_i, y_i)\}_{i=1}^{B}$. A forward pass produces predictions $\hat{y}_i = f_\theta(x_i)$ and a batch loss $\mathcal{L}_\mathcal{B} = \frac{1}{B}\sum_{i\in\mathcal{B}} \mathcal{L}(\hat{y}_i, y_i)$. Backpropagation then applies the chain rule to compute $\nabla_\theta \mathcal{L}_\mathcal{B}$ efficiently from output back to inputs [18, 19]. Finally, an optimizer (e.g. SGD or Adam) updates the parameters θ.

Geometric picture. Figure 3.7 depicts level sets (contours) of a loss function over a 2D slice of parameter space (θ_1, θ_2). Each closed curve ("circle") is a set of parameters with equal loss. Moving inward toward the center means lower loss. The negative gradient $-\nabla_\theta \mathcal{L}$ points in the direction of steepest local decrease and thus indicates the shortest way to the next lower contour. A plain SGD step $\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta \mathcal{L}_\mathcal{B}(\theta^{(t)})$ follows this direction with step size η (learning rate). Momentum and Adam modify the raw gradient to smooth or precondition the step [17, 20].

What each element means.

- *Contours (circles):* sets $\{\theta : \mathcal{L}(\theta) = c\}$ for different constants c. Smaller contours correspond to lower loss values.
- *Gradient vector:* $\nabla_\theta \mathcal{L}$ is orthogonal to the contour and points toward higher loss; $-\nabla_\theta \mathcal{L}$ points toward lower loss.
- *Learning rate η:* step length. Too small η crawls; too large can overshoot or diverge.
- *Momentum/Adam:* accumulate smoothed gradient statistics to take longer steps along gently sloped directions and shorter steps along steep axes, which helps on the elongated contours.

From this toy picture back to CNNs. In our segmentation networks, θ contains millions of weights across convolutional filters, normalization parameters, and biases. Backpropagation propagates errors from the pixel-wise loss (BCE+Dice) through the decoder, skip connections, and encoder to compute $\nabla_\theta \mathcal{L}$. Adam then applies an adaptive, per-parameter step that typically converges faster and more stably than plain SGD for our setting.

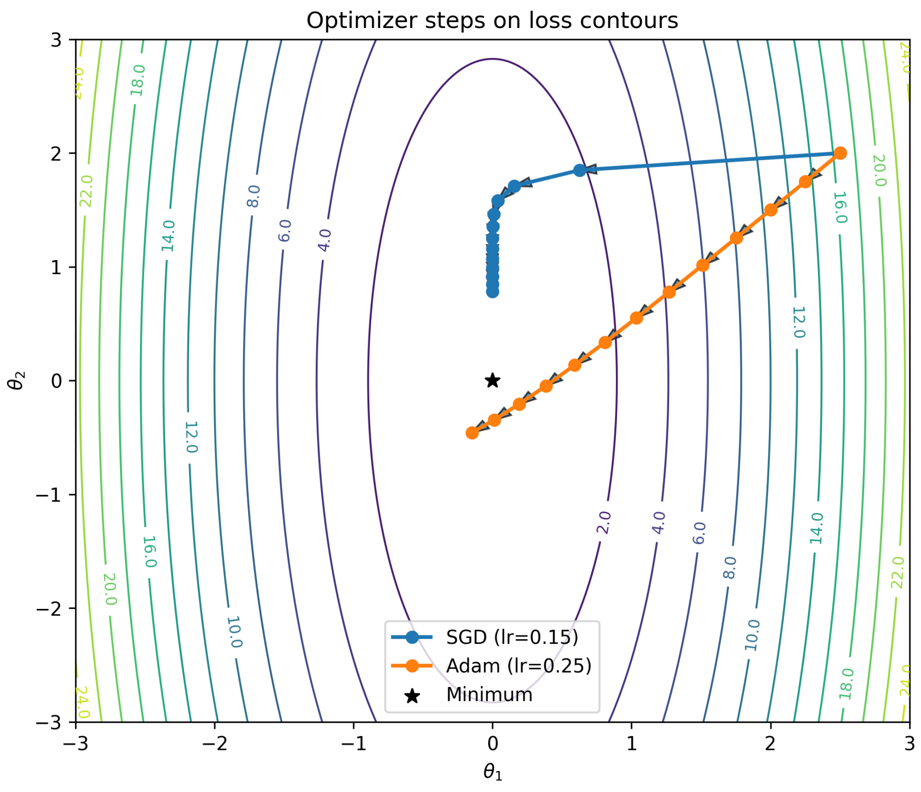


Figure 3.7: Loss contours (level sets) and optimizer trajectories on a 2D quadratic loss. Each curve is an equal-loss boundary; moving inward lowers the loss. Arrows show parameter updates for several steps of SGD (blue) versus Adam (orange) from the same start point.

3.5.6. Training On Data: Batches, Iterations, and Stopping Criteria

Previously, we have illustrated a single parameter update. However, CNNs are trained on large datasets that cannot be processed all at once due to hardware memory limits. Instead, training proceeds in mini-batches, with multiple passes through the dataset until convergence [19, 21]. To do so, we must first split the dataset.

Dataset Splits

The full dataset is typically divided into:

- **Training set:** used to update the model parameters.
- **Validation set:** used to monitor generalization and tune hyperparameters.
- **Test set:** unseen data used only for final evaluation.

Batch and Mini-Batch

Suppose the training set has N samples. Instead of computing gradients on all N at once, we divide them into *mini-batches* of size B:

$$\{(x_{i_1}, y_{i_1}), \dots, (x_{i_B}, y_{i_B})\}.$$

For each mini-batch, we run a forward pass, compute the loss, and update the parameters. This strikes a balance between the noisy updates of stochastic gradient descent (where $B = 1$) and the high memory cost of full-batch gradient descent ($B = N$). Typical mini-batch sizes range from 16 to 256, depending on dataset size and GPU memory.

Iteration and Epoch

One gradient update on a single mini-batch is an iteration. Processing all N samples once (i.e., all N/B mini-batches) is called an epoch. Training usually involves many epochs (tens to hundreds or thousands), during which the optimizer gradually reduces loss and improves accuracy.

Stopping Criteria and Overfitting

Previously, we introduced the idea of iteratively updating parameters through gradient-based optimization. In practice, it is not possible train indefinitely: at some point, the network begins to memorize the training data instead of learning generalizable patterns. This phenomenon is known as overfitting.

Training vs. Validation Loss. A standard way to monitor training progress is by plotting both training and validation losses across epochs. Figure 3.8 illustrates the typical behavior [19, 22]:

- During early epochs, both training and validation loss decrease, as the network learns useful representations.
- After a certain point, training loss continues to decline, but validation loss starts to rise. This indicates that the model is fitting noise and peculiarities of the training set, rather than general structure [23].
- The epoch corresponding to the minimum validation loss is usually the best checkpoint to select. Training beyond this point worsens generalization [22].

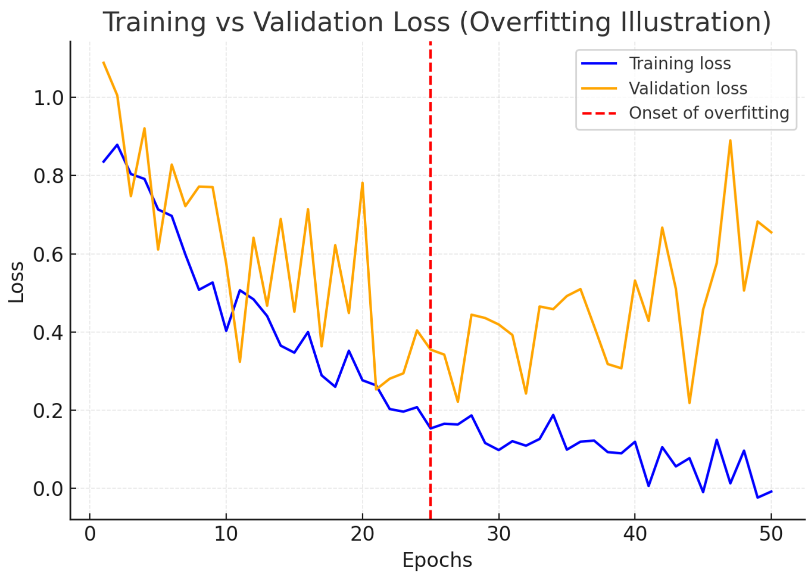


Figure 3.8: Illustration of overfitting during CNN training [19]. Training loss (blue) decreases continuously. Validation loss (orange) decreases initially but begins to rise once the model overfits. The optimal stopping point is often taken at the minimum of the validation curve.

Epochs, Batches, and Iterations. Training data are rarely processed all at once due to hardware limitations. Instead:

- A **batch** is a subset of the dataset passed through the network in one forward and backward pass.
- A **mini-batch** refers to this same concept, typically containing 16–256 samples depending on GPU memory.
- An **iteration** is one update step based on a single mini-batch.
- An **epoch** is a complete pass through the entire dataset, i.e., $\frac{\text{dataset size}}{\text{batch size}}$ iterations.

Thus, training usually spans multiple epochs, where the optimizer updates weights iteratively across mini-batches until validation performance stabilizes [19].

Why Stopping Criteria Matter. Without a stopping rule, a model may continue reducing training loss indefinitely, while its performance on unseen data deteriorates. Stopping criteria ensure a balance between underfitting (too few epochs, model fails to capture structure) and overfitting (too many epochs, model memorizes noise). Common approaches include:

- **Early stopping:** halt training when validation loss stops improving for k consecutive epochs [22].
- **Maximum epochs:** a hard cap on the number of epochs, often combined with early stopping.
- **Regularization-aware monitoring:** techniques such as dropout [24] or weight decay that reduce the risk of overfitting, thereby allowing longer training before validation loss diverges.

Dataset Size and Generalization. The balance between training time and generalization is influenced by dataset size. With small datasets, overfitting tends to occur earlier, requiring stronger regularization and aggressive early stopping. Larger datasets allow longer training, as each epoch exposes the network to more diverse samples, which delays overfitting [23, 19].

In summary, monitoring the gap between training and validation loss provides a consistent way to decide when to stop training. The optimal model is not the one with the lowest training loss, but the one that achieves the lowest validation loss, ensuring the best generalization to unseen data.

Putting It Together

Training a CNN can thus be summarized as:

1. Randomly initialize parameters.
2. For each epoch:
   (a) Shuffle the training data and divide into mini-batches.
   (b) For each mini-batch:
      i. Perform a **forward pass** to compute predictions.
      ii. Calculate the **loss function** comparing predictions with ground truth.
      iii. Apply **backpropagation** to compute gradients of the loss w.r.t. all parameters.
      iv. **Update parameters** using the chosen optimizer (e.g. SGD or Adam), scaled by the current learning rate η.
   (c) Evaluate the model on the validation set to monitor generalization.
3. Stop according to the chosen criterion (e.g. early stopping, maximum epochs).

In summary, each iteration moves the parameter vector one step closer to a minimum of the loss surface, guided by gradients and scaled by the learning rate. Repeating this cycle across many mini-batches and epochs gradually refines the network until it achieves parameters that not only fit the training data but also generalize to unseen inputs.

3.5.7. Regularization and Generalization

Finally, regularization strategies are essential to prevent overfitting. Key approaches include:

- **Weight decay:** penalizes large parameter values to discourage overconfident models.
- **Data augmentation:** increases the effective diversity of the dataset by introducing variations in the input images.
- **Early stopping:** halts training once validation performance stops improving.
- **Architectural techniques:** elements such as Batch Normalization and skip connections improve generalization by stabilizing gradients and optimizing training dynamics.

4
Segmentation Architectures Used in This Thesis

4.1. Segmentation as Dense Pixel-Wise Prediction

Convolutional neural networks (CNNs) form the backbone of most modern computer vision systems. They can be applied to a wide range of tasks, from simple image-level classification to more fine-grained analyses (Figure 4.1):

- **Classification:** predict a single label for the whole image (e.g., “cat”).
- **Classification + Localization:** additionally provide a bounding box indicating where the object is.
- **Object Detection:** detect multiple objects of different classes and output bounding boxes for each.
- **Instance Segmentation:** delineate the precise pixel-wise boundaries of each object instance.

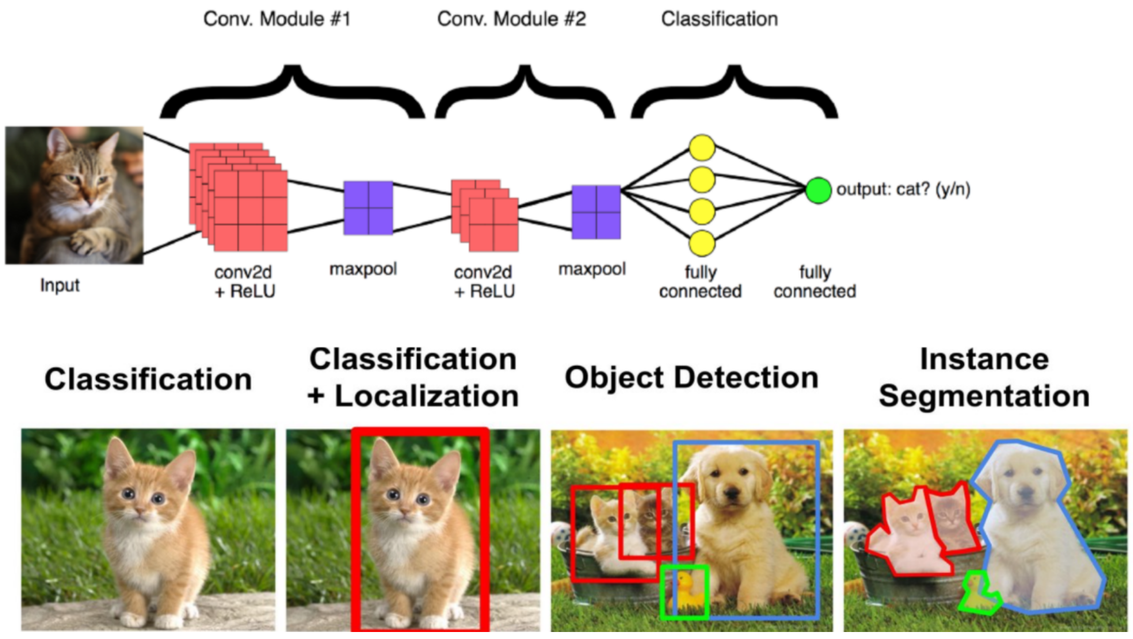


Figure 4.1: Illustration of CNN tasks in computer vision. A CNN can classify entire images, localize objects, detect multiple instances, or perform dense pixel-wise segmentation. Biomedical image analysis typically requires the latter, as subtle structures must be localized with pixel-level precision. Image from [25]

For biomedical applications such as collagen fiber analysis in engineered heart tissues (EHTs), dense pixel-wise segmentation is what is needed. Unlike classification or detection, which provide only coarse information, segmentation assigns a label to every pixel. This enables quantitative measurement of fiber lengths, connectivity, and orientations metrics that are critical for linking microscopic structure to macroscopic tissue function.

Patch-Based vs. Pixel-Based Segmentation. Before the development of encoder–decoder networks, one common strategy was to train CNNs in a patch-based manner: a small window (patch) around each pixel was extracted and classified independently. While effective, this approach was computationally inefficient and often failed to capture long-range context. Modern architectures like U-Net [26] instead perform pixel-to-pixel segmentation: the entire image is processed end-to-end, and the network directly outputs a segmentation mask of the same resolution (Figure 4.2). This not only accelerates inference but also allows the model to utilize both local and global spatial context.

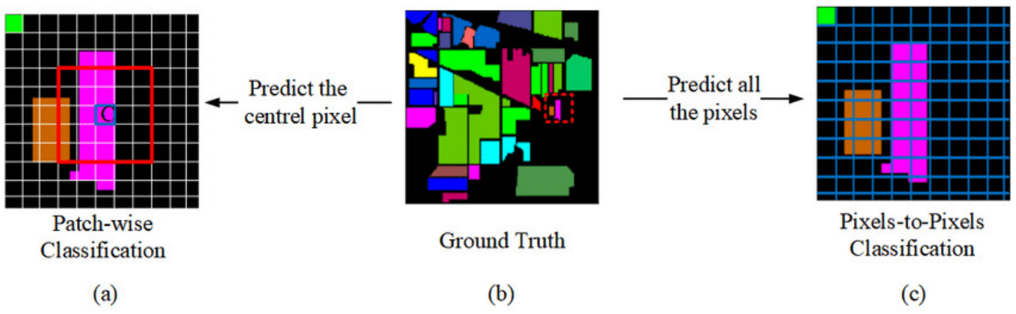

C
Predict the centrel pixel
Predict all the pixels
Patch-wise Classification
Ground Truth
Pixels-to-Pixels Classification
(a)
(b)
(c)

Figure 4.2: Patch-wise vs. pixel-wise segmentation. In patch-based approaches, small windows around pixels are classified independently, which is inefficient and lacks global context. Pixel-to-pixel architectures like U-Net process the entire image in one forward pass, producing dense predictions at the original resolution. Image from [27]

Why U-Net? The U-Net architecture was designed specifically to address the challenges of biomedical segmentation, where:

- Accurate delineation of thin structures (fibers, membranes, vessels) is essential.
- Annotated data are often limited, requiring efficient use of available information.
- Pixel-wise labels, not just image-level predictions, are needed for quantitative analysis.

U-Net extends CNN's by implementing an encoder–decoder structure with skip connections, enabling it to combine global context with fine-grained localization [26]. In the following sections, we will describe this architecture in detail and introduce the variants used in this thesis.

4.2. From CNNs to U-Nets

While CNNs excel at extracting features, their canonical form was designed for classification, with repeated downsampling steps that collapse spatial detail into global feature vectors. Such architectures are not effective for segmentation, where precise spatial correspondence between input and output must be preserved.

The U-Net architecture [26] was introduced specifically for biomedical image segmentation. It retains the strengths of CNNs for feature extraction, but augments them with a symmetric encoder–decoder structure and long-range skip connections. This design enables the network to capture both global context (via encoding) and fine detail (via skip pathways), making it highly effective on datasets where annotated images are scarce but high-resolution pixel-level accuracy is essential.

Figure 4.5 illustrates the canonical U-Net, showing how contracting and expanding paths are connected by skip links.

4.3. Encoder-Decoder with Skip Connections

The encoder–decoder principle is the basis of the UNet structure:

- The **encoder** progressively downsamples the input image using convolutions and pooling. This aggregates information over larger receptive fields, capturing context and global structure.
- The **decoder** then upsamples the compressed representation back to the original resolution using transposed convolutions or interpolation layers, reconstructing spatial detail.
- **Skip connections** pass intermediate encoder features directly to the decoder at matching resolutions. This preserves high-frequency detail, ensuring that thin or small structures (such as fibers) are not lost during downsampling.

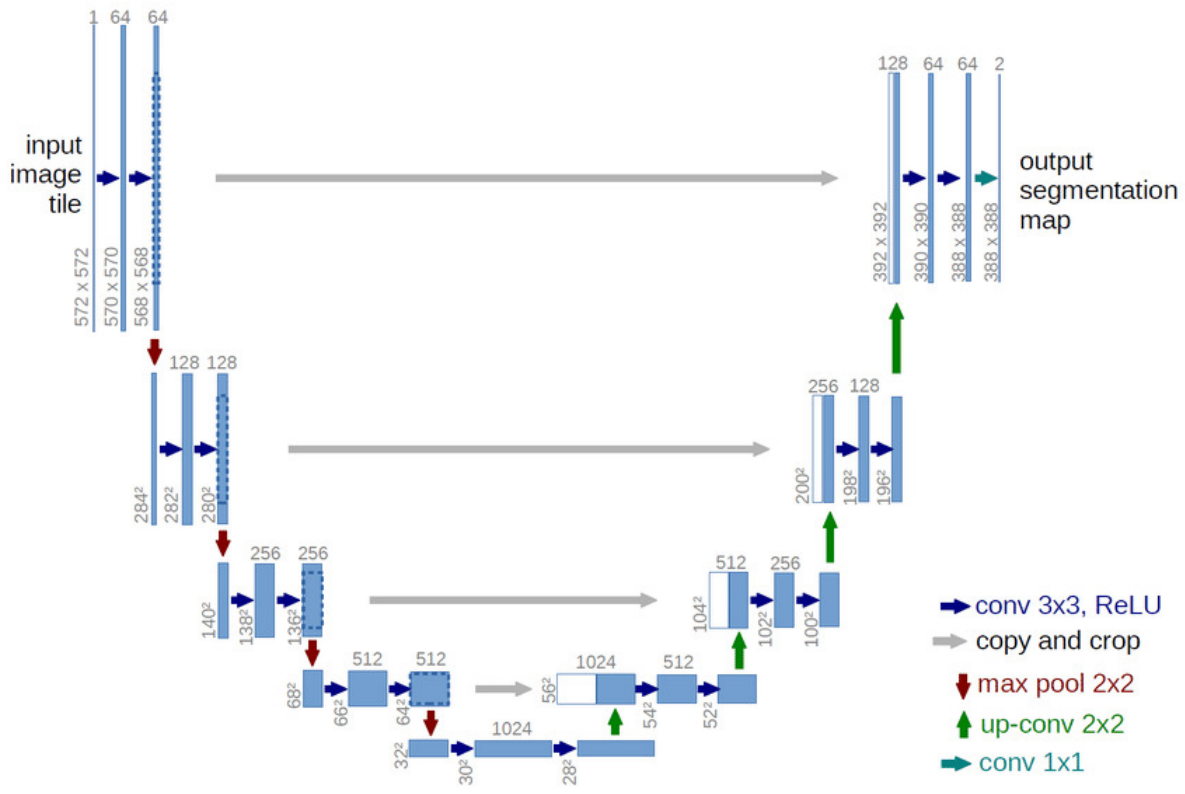


Figure 4.3: Encoder–decoder architecture of the canonical U-Net with skip connections. The encoder compresses the input into high-level representations, while the decoder reconstructs dense predictions. Skip connections transfer spatial detail directly to the decoder, improving localization of thin structures. Image from [26]

4.3.1. Encoder-Decoder Walkthrough

To make the role of pooling and skip connections concrete, we illustrate the encoder–decoder process using a toy example. The input is a 5×5 grid with black pixels representing zeros and a single gray dot in the center. For all the images below, the lighter the color, the higher the value. We apply a 3×3 convolutional kernel, followed by a nonlinearity (ReLU), pooling to a 1×1 bottleneck, and finally decoding back to the original 5×5 size. Figure 4.4 shows the result with and without a skip connection.

What each panel shows.

1. **Input (5$\times$5)**: black background (0) with a single bright pixel (1.0) in the center.
2. **Kernel (3$\times$3)**: the filter used to detect local structure. Lighter shading indicates higher weights.
3. **Convolution (3$\times$3)**: local weighted sums of the input, producing a 3×3 feature map.
4. **ReLU (3$\times$3)**: keeps positive activations, zeroes out negative responses.
5. **Max-pooling (1$\times$1)**: collapses the 3×3 activations to a single bottleneck value, discarding almost all spatial information.

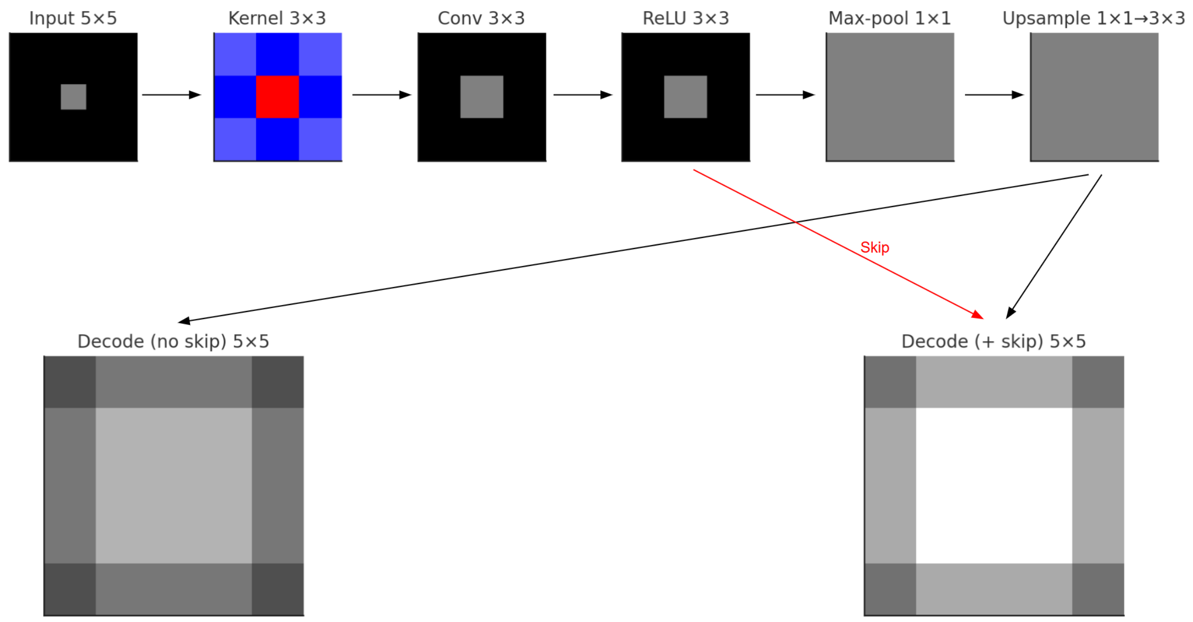


Figure 4.4: Toy encoder→bottleneck→decoder with and without a skip connection. Black corresponds to 0 and lighter shades indicate stronger activations. The arrows illustrate how information flows through the encoder (top row) and decoder (bottom row). The red dashed arrow highlights the skip connection: activations from the encoder's 3×3 ReLU feature map are passed directly to the decoder at the same scale, supplementing the upsampled bottleneck. Without this skip (bottom left), the reconstruction is blurred because the 1×1 bottleneck loses most of the fine detail. With the skip (bottom right), high-frequency information is restored, producing a noticeably sharper reconstruction.

6. **Decoder (5×5)**: reconstruction by upsampling and filtering. Without a skip, detail is blurred. With a skip connection, the 3×3 encoder features are added back into the decoding process, restoring sharpness and preserving fine structures.

Why skips help. Downsampling increases context but discards precise locations of edges and thin structures. Skip connections copy high-resolution encoder features directly to the decoder at the same spatial scale. These features carry the missing fine detail , allowing the decoder to combine global context (from the bottleneck) with sharp localization (from the skip). This is the rationale behind U-Net [26] and its variants for biomedical segmentation, where boundaries and filaments are thin and easily lost by pooling.

4.4. Architectures Used in This Thesis

We now describe the three segmentation architectures evaluated in this work, all of which build upon the encoder–decoder foundation but have additional features. Each architecture addresses the same core task: producing a dense, pixel-wise segmentation map, but they differ in how they preserve detail, filter information, or fuse features at multiple scales.

4.4.1. U-Net (Ronneberger et al., 2015)

The U-Net [26] is a landmark architecture designed for biomedical segmentation. The "U" shape has a contracting path (encoder) and an expansive path (decoder). At each downsampling step the number of channels doubles, while at each upsampling step it halves. Crucially, U-Net concatenates encoder features to decoder features at the same resolution (skip connections), so localization (edges, thin fibers) is preserved while global context is retained.

In Figure 4.5, the encoder (left arm) progressively compresses the input into higher-level representations, while the decoder (right arm) upsamples to restore spatial detail. The horizontal skip connections transfer fine-grained encoder features directly into the decoder, preventing loss of localization.

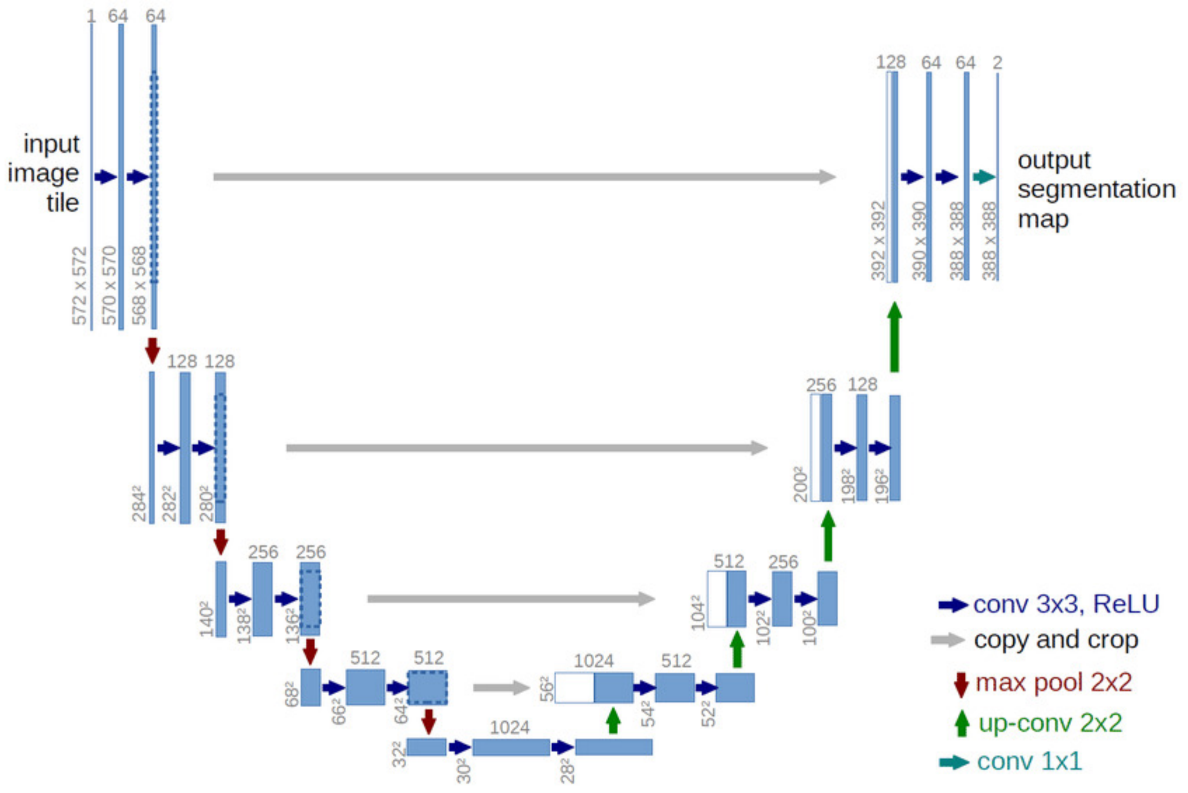
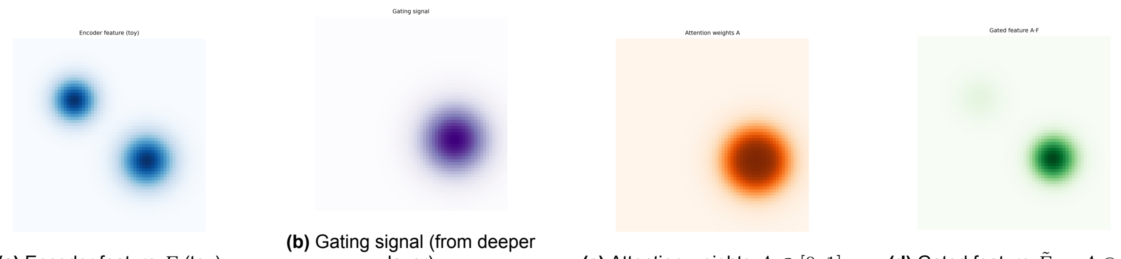


Figure 4.5: Canonical U-Net architecture [26]. Skip connections concatenate encoder features into the decoder to preserve spatial detail crucial for thin structures.

4.4.2. Attention U-Net (Oktay et al., 2018)

Skip connections are helpful, but they may also forward irrelevant background signal. **Attention U-Net** [28] introduces attention gates that learn to apply a weight to the encoder features before they are fused into the decoder.

What is an Attention Gate? An attention gate computes a spatial weight map $A \in [0,1]^{H \times W}$ using (i) encoder features and (ii) a gating signal from deeper layers. The weight map A highlights relevant regions (e.g., fibers) and suppresses irrelevant ones. The gated feature $\tilde{F} = A \odot F$ is then passed along the skip.



(a) Encoder feature F (toy) **(b)** Gating signal (from deeper layer) **(c)** Attention weights $A \in [0,1]$ **(d)** Gated feature $\tilde{F} = A \odot F$

Figure 4.6: Toy illustration of an attention gate in Attention U–Net [28]. Panel (a): encoder feature map with two activations (blue; darker = stronger response, lighter = weaker/zero). Panel (b): gating signal from deeper layers (purple; darker = stronger cue). Panel (c): attention weights A computed from (a) and (b) (orange; darker = higher weight), highlighting the relevant region. Panel (d): elementwise gating $\tilde{F} = A \odot F$ (green) suppresses the irrelevant blob (top-left) and preserves the relevant one (bottom-right). The gated feature $\tilde{F}$ is then passed through the skip connection, reducing background clutter while keeping relevant structure. Colors indicate relative magnitude only (darker = higher).

Attention Demo. To make this concept concrete, Figure 4.6 shows a simple illustration. The encoder feature contains two blobs, and the gating signal marks only one as relevant. The attention weights A

are high at that location, and the final gated feature retains only the important blob. This is analogous to how the Attention U-Net learns to suppress clutter while keeping fibers.

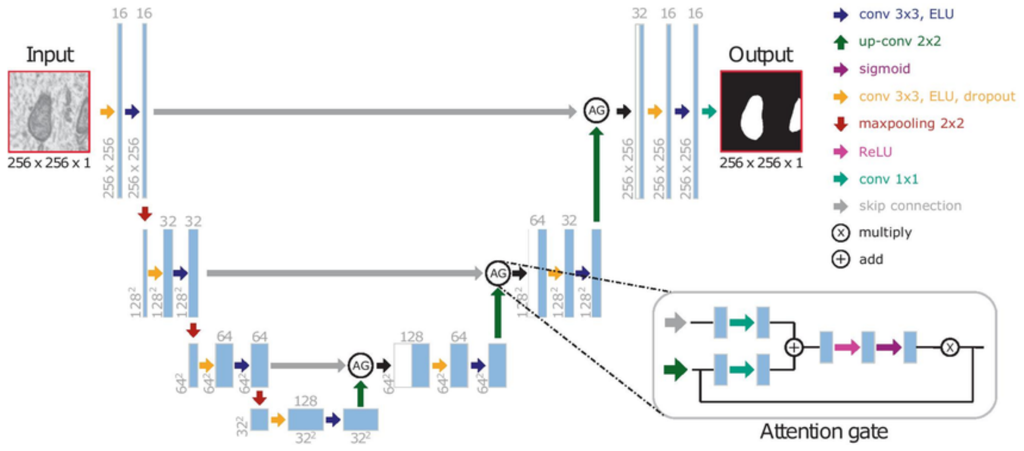


Figure 4.7: Canonical Attention U-Net architecture [28]. The encoder path (left, blue blocks) compresses the input into high-level features, while the decoder path (right, green blocks) restores spatial detail through upsampling. Horizontal skip connections (gray arrows) transfer encoder features into the decoder, but here each skip passes through an **attention gate** (orange blocks). These gates compute a spatial weight map that suppresses irrelevant background regions and highlights important structures before fusion with the decoder.

Attention in Practice. Figure 4.7 shows the canonical Attention U-Net diagram from [28]. Attention gates (AG) are placed on the skip pathways, where the green arrows mark modulation of encoder features by gating signals from deeper context. This allows the decoder to highlight fibers while ignoring clutter. In practice, this improves segmentation quality in challenging biomedical images where fine boundaries must be distinguished from noise.

4.4.3. U-Net++ (Zhou et al., 2018)

U-Net++ [29] enhances the original U-Net by introducing nested, dense skip pathways. The motivation is that in a plain U-Net, encoder features and decoder features at the same resolution may still be very different in "semantics." For example, the encoder may contain low-level edges or textures, while the decoder expects features already aligned with segmentation masks. Passing them directly can cause a mismatch.

Nested Skips. Instead of one direct skip, U-Net++ inserts intermediate convolutional blocks along the pathway. These refine the encoder output step by step before it reaches the decoder. Each intermediate node acts like a small translator, narrowing the semantic gap so that when features finally arrive at the decoder, they are more useful for precise segmentation.

Dense Connectivity. In addition, U-Net++ connects these intermediate nodes laterally across scales (dense links). This allows information to flow not only vertically (encoder to decoder) but also horizontally (between refinements at the same resolution). As a result, features are combined from multiple depths and receptive fields, which is especially beneficial for faint, thin, or overlapping fibers where local and global context must be fused.

Canonical U-Net++ Figure. Figure 4.8 shows the canonical U-Net++ diagram. The top-left subfigure (a) is the plain U-Net with direct skips, while (b) shows U-Net++ with nested and dense skip connections. Notice how each decoder stage now receives features that have been progressively refined through multiple intermediate nodes, rather than raw encoder features. This refinement reduces the semantic gap and improves final segmentation accuracy. The figure also shows some other variations of the unet++, but these are unused in this thesis and are here simply to showcase them.

Toy Illustration. Figure 4.9 illustrates this concept in a simplified form. Blue circles represent feature maps, while red arrows indicate the main encoder and decoder path. Blue arrows show the additional

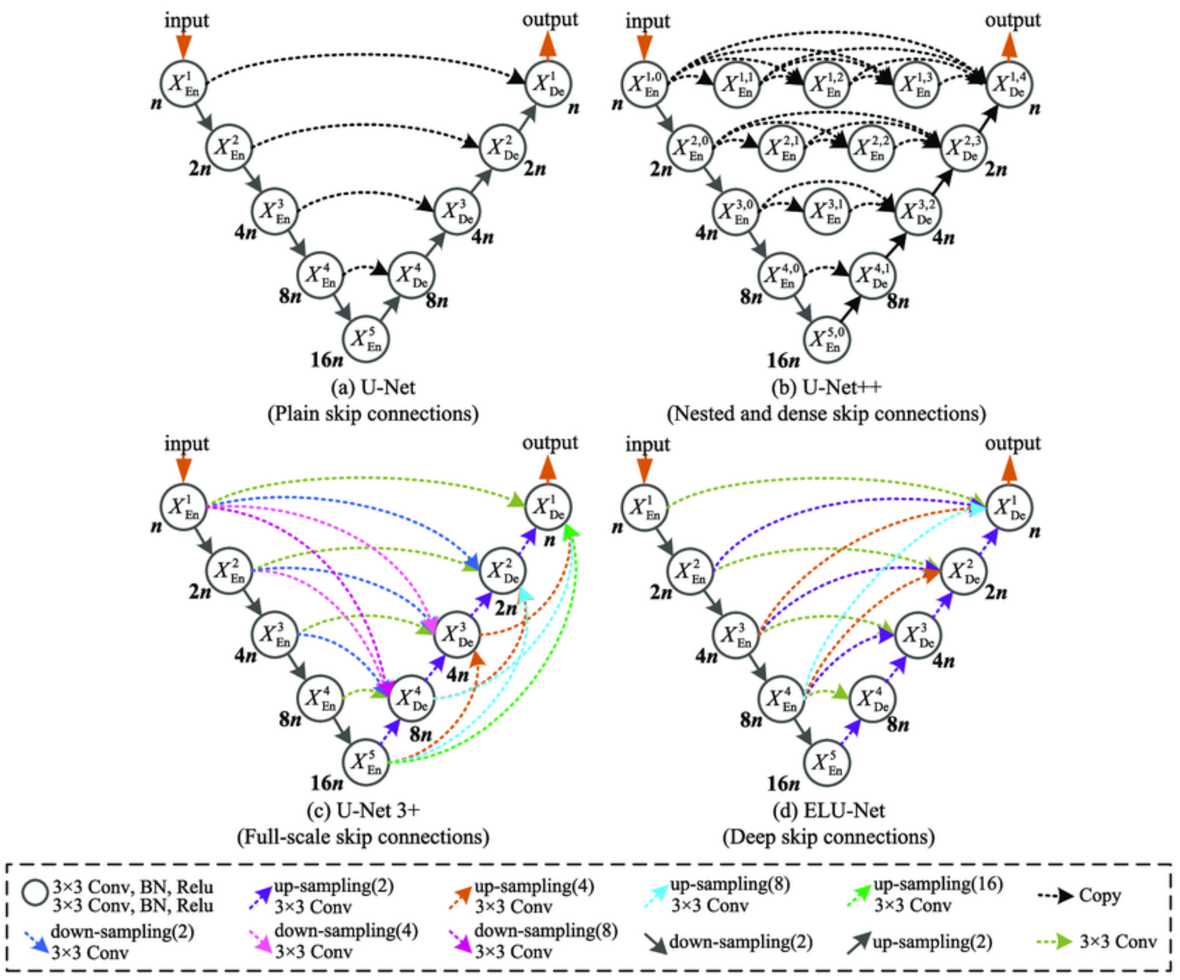


Figure 4.8: Canonical U-Net++ [30]. (a) Plain U-Net skip connections. (b) U-Net++ nested and dense skip pathways: intermediate nodes refine encoder features, while dense lateral links fuse information across scales. This design improves multi-scale feature fusion and localization of thin, complex structures.

nested connections that refine skip features before they reach the decoder, and red dashed lines depict dense lateral links. Together, these create a grid of intermediate nodes that gradually refine information rather than sending it in a single step. Conceptually, U-Net++ can be seen as building several short bridges across a river rather than a single long one, allowing information to flow more smoothly and reliably.

4.4.4. U-Net++ with Pretrained Backbones

While U-Net++ already improves segmentation through nested and dense skip connections, its performance can be further enhanced by using a pretrained backbone as the encoder.

What is Pretraining? Pretraining refers to initializing part of a model with weights learned on a large, generic dataset such as ImageNet [31]. These datasets contain millions of images across many categories, so the network learns general-purpose features in its early layers:

- Lower layers capture edges, corners, and simple textures.
- Intermediate layers capture object parts or repeating patterns.
- Higher layers capture more abstract concepts, such as shapes or categories.

By reusing these pretrained weights, we avoid training from scratch and leverage the “knowledge” already embedded in the backbone.

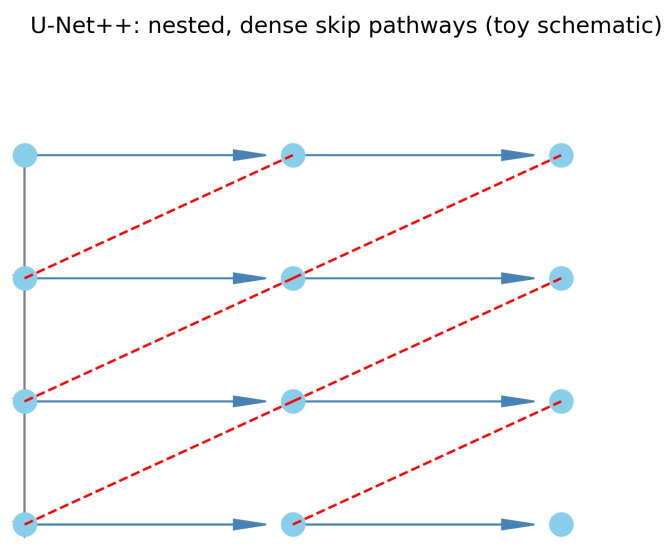


Figure 4.9: Toy schematic of U-Net++ nested skips. Blue nodes are intermediate refinements; gray arrows the main backbone; blue arrows the nested skip hops; red dashed lines the dense lateral links. These refinements bridge the gap between encoder and decoder features, improving multi-scale fusion and localization.

How Backbones Fit into U-Net++. In U-Net++, the encoder path can be replaced by a pretrained CNN backbone (e.g., ResNet, VGG, or EfficientNet). Instead of random initialization, the encoder convolutions start with weights from pretraining. The decoder path and skip refinements are then trained specifically for the biomedical segmentation task. This approach combines the strong representation power of pretrained models with the multi-scale refinement of U-Net++.

Fine-Tuning. During training, there are two common strategies:

- **Frozen backbone:** the pretrained encoder weights are kept fixed, and only the decoder and skip pathways are trained. This is useful when the dataset is small, to avoid overfitting.
- **Fine-tuned backbone:** the pretrained encoder is further trained alongside the decoder, but often with a smaller learning rate. This allows the model to adapt general features to the biomedical domain while retaining its prior knowledge.

Why This Helps. Biomedical datasets are often limited in size, so training deep networks from scratch can lead to poor generalization. Pretrained backbones provide a strong initialization, speeding up convergence and improving accuracy, particularly for subtle features like faint or thin fibers. In practice, this approach consistently outperforms randomly initialized models, especially when combined with data augmentation and regularization.

Summary

Pixel-wise encoder-decoder models are the backbone of biomedical segmentation.

- **U-Net** preserves spatial detail by directly concatenating encoder features into the decoder via skip connections, making it effective even with limited data.
- **Attention U-Net** filters these skips with attention gates, ensuring that only important, task-relevant features (e.g., fibers) reach the decoder while suppressing background noise.
- **U-Net++** refines skip information with nested, densely connected pathways that gradually bridge the semantic gap between encoder and decoder, improving delineation of thin and overlapping structures.

- **U-Net++ with Pretrained Backbones** further enhances performance by initializing the encoder with weights learned from large-scale datasets such as ImageNet. This leverages general visual features, reduces training time, and improves generalization when biomedical datasets are small.

Together, these architectures represent the different techniques used: direct skips, gated skips, refined multi-scale skips, and pretrained feature hierarchies. Each step addresses the central challenge of combining global context with precise localization for accurate fiber segmentation. Pixel-wise encoder-decoder models are the backbone of biomedical segmentation. U-Net preserves detail via direct skips, Attention U-Net filters those skips so the decoder sees specific content, U-Net++ further refines skip information with nested, densely connected pathways, improving recovery of thin and complex structures.

5 Datasets and Preprocessing

5.1. Real-World EHT Confocal Volumes

The volumetric confocal datasets used in this thesis were acquired using Imaris, a proprietary microscopy acquisition and analysis suite. Since Imaris file formats are not directly compatible with Python-based image processing workflows, the volumes were exported as multi-page TIFF stacks. TIFF was chosen over JPEG as it preserves the full dynamic range and avoids compression artifacts that would compromise quantitative analysis.

The dataset consists of three 3D images, each with dimensions of approximately $9300 \times 5400 \times 48$ (width $\times$ height $\times$ slices). In practice, many of the z-slices contained little or no visible signal, so the effective depth was closer to 25 slices per volume. Across the three stacks, only one was manually annotated: three representative slices were traced in Photopea using a pen tool to create binary masks of fibers. These annotations are sparse and were produced by a novice labeler, but they provide a limited ground truth reference for testing segmentation models.

The annotation exercise also highlights the challenges of this dataset. Many slices appear nearly empty, and in those that contain signal, fibers are faint and difficult to trace consistently. This subjectivity and sparsity reinforce the need for automated segmentation methods that can reliably capture fiber structures at scale.

5.2. Secondary Synthetic Collagen Dataset

The secondary dataset originates from the work of Park et al. [32], who proposed a deep-learning pipeline for collagen fiber centerline tracking in fibrotic tissues. A key innovation of their study was the creation of a synthetic training dataset using a variational autoencoder (VAE).

In their approach, binary label maps of collagen fibers were first generated, representing ground-truth centerlines and fiber structures. A VAE was then trained to translate these binary maps into realistic second-harmonic generation (SHG)-like images, capturing the noise patterns, intensity variations, and textural appearance of actual microscopy acquisitions. This procedure allowed the authors to generate large volumes of paired image–label data, where the underlying “ground truth” was fully controlled.

The final dataset consisted of 1200 training images and 300 test images, each accompanied by a pixel-accurate label. Although the images are synthetic, they closely resemble real SHG collagen imaging, making them suitable for supervised training of segmentation models. For this thesis, this dataset provided the necessary scale and annotation quality to pretrain deep learning architectures before applying them to the smaller, noisier, and more complex engineered heart tissue (EHT) confocal images.

It should be noted, however, that while the synthetic dataset captures many statistical and textural features of fibrous tissues, it inevitably lacks the structural complexity observed in real EHTs and thus the real EHT images. Fibers in synthetic images tend to be more homogeneous, better aligned, and less entangled compared to those in EHT slices, which display dense, irregular and heterogeneous organizations. Despite this limitation, the Park dataset provides a strong baseline for initial training and benchmarking, to evaluate future potential in the field.

5.3. FFT Bandpass Filtering

Preprocessing was conducted using frequency-domain bandpass filtering. As described in detail in Section 2.3.2, Fourier transforms allow an image to be represented in terms of its spatial frequency components. By applying a mask in the frequency domain and reconstructing the slice with the inverse transform, we can isolate the frequencies corresponding to fiber boundaries, while suppressing both low-frequency illumination gradients and high-frequency noise.

Compared to sequential highpass–lowpass filtering, this FFT-based approach provides finer control over the retained frequency band and is more robust to the variability of real confocal data. For this reason, it was adopted as the standard preprocessing step throughout the experiments in this thesis.

5.4. Alternative: Highpass–Lowpass Filtering

We briefly considered a simpler highpass–lowpass filtering approach, where an image is convolved with Gaussian kernels to remove unwanted frequencies.

Formally, given an image $f(x, y)$ and a Gaussian kernel $g_\sigma(x, y)$ with standard deviation σ, a lowpass filter is defined as

$$f_{\text{low}}(x, y) = (f * g_\sigma)(x, y),$$

where $*$ denotes convolution. This operation removes high-frequency detail (sharp edges, noise) while retaining smooth background variations.

A corresponding highpass filter can be obtained by subtracting the lowpass image from the original:

$$f_{\text{high}}(x, y) = f(x, y) - f_{\text{low}}(x, y).$$

This emphasizes rapid intensity changes such as edges and fine texture. By chaining lowpass and highpass filters with different σ, we create a bandpass filter that maintains the features of the desired structure.

While conceptually straightforward, this method provides less precise control over the retained frequency band. The choice of σ directly determines the effective cutoff frequency, but this relationship depends on the image resolution and can be difficult to tune consistently across datasets. Moreover, unlike FFT-based filtering, the frequency response cannot be directly visualized.

In contrast, FFT bandpass filtering (Section 2.3.2) allows us to inspect the magnitude spectrum of an image, define exact low- and high-cut values, and explicitly mask frequencies outside the desired range. This makes FFT filtering more transparent and more adaptable to the heterogeneity of confocal data.

A short comparison confirmed that FFT-based filtering produced clearer and more interpretable results for fiber segmentation. For this reason, all downstream experiments in this thesis employ FFT bandpass filtering as the primary preprocessing step.

5.5. Denoising Experiments for Confocal Imaging

This supplementary section documents the denoising techniques explored in addition to the FFT band-pass used in the main pipeline. The goal was to test whether classical deconvolution and intensity-domain filtering could further improve fiber visibility prior to segmentation. In practice, with our EHT data, these approaches did not produce consistent improvements beyond the FFT band-pass, but they are promising directions for future work.

5.5.1. Deconvolution Approaches (PSF-Based)

Confocal images are well modeled as a convolution of the underlying object with the point-spread function (PSF) plus noise. We trialed two classical deconvolution techniques with synthetic Gaussian PSFs (2D and 3D) because the true PSF was unavailable.

Tikhonov–Miller (Wiener-like) Deconvolution. Let g be the observed image, h the PSF, and f the latent object. In the Fourier domain,

$$\widehat{F}(u,v) = \frac{\overline{H}(u,v)}{|H(u,v)|^2 + \lambda}\,\widehat{G}(u,v),$$

where $\lambda > 0$ is the Tikhonov/Miller regularization parameter. This stabilizes the inversion of small $|H|$ but introduces bias; too small λ amplifies noise (ringing), too large over-smooths fibers. With approximate Gaussian PSFs (e.g. 21×21, $\sigma_y = \sigma_x \in [2,4]$), we saw local sharpening but also noise amplification on EHT slices, and no consistent benefit downstream.

Richardson–Lucy (RL) Deconvolution. RL performs iterative, non-negative deconvolution assuming Poisson statistics:

$$f^{(k+1)} = f^{(k)}\left[\left(\frac{g}{f^{(k)} * h}\right) * h^{\star}\right],$$

with $h^{\star}$ the flipped PSF. We tested 2D RL on slices and 3D RL on small volumes using Gaussian PSFs (e.g. 21^3 with anisotropic $\sigma_z > \sigma_{x,y}$). RL can sharpen fibers when the PSF is accurate, but with a mismatched/guessed PSF we observed grain amplification and staircase artifacts after 20–100 iterations. Even when visually crisper, segmentations (Dice/IoU) did not improve reliably over the FFT baseline.

5.5.2. Practical Notes and Failure Modes

- **PSF mismatch:** Using a synthetic Gaussian PSF (no instrument calibration) limits deconvolution benefits; errors in width/anisotropy translate into ringing or over-sharpening.
- **Noise amplification:** Both TM and RL can boost salt-and-pepper–like speckle, making thin fibers harder to threshold robustly without extra denoisers.
- **Parameter sensitivity:** TM's λ, RL's iteration count, and PSF size/sigmas require per-volume tuning. Settings that help one slice often harm another.

5.5.3. Outcome

Across EHT slices, the FFT band-pass (2.3.2) consistently delivered the best trade-off: it removes low-frequency illumination and high-frequency sensor noise while preserving mid-frequency fiber edges. Deconvolution and intensity-domain high-pass sometimes improved local contrast, but at the cost of intensified noise. Given our unknown PSF and heterogeneous volumes, we therefore standardized on FFT band-pass for all downstream experiments.

6 Engineering, Reproducibility, and Ethics

6.1. Engineering and Reproducibility

This thesis puts an emphasis on engineering clarity and reproducibility. All preprocessing steps, model architectures, training procedures, and evaluation metrics have been described in detail to enable full replication of the experiments. In addition, the complete codebase used in this thesis will be made publicly available (`in this GitHub repository`), ensuring that others can build on the pipeline without ambiguity. Anyone interested in extending or applying this work are welcome to contact the author for clarification or collaboration.

6.2. Ethics and Limitations

Beyond technical considerations, biomedical applications carry ethical and scientific responsibilities.

Hallucinations in medical imaging. Deep networks may produce hallucinated structures, i.e. plausible-looking but nonexistent fibers. In medical contexts, such outputs could mislead downstream analysis or clinical decision-making. We explicitly report such limitations by combining quantitative metrics with qualitative human inspection, highlighting both successes and failure cases.

Dataset mismatch. The primary dataset consists of engineered heart tissue (EHT) confocal volumes, while the secondary dataset (Park et al. [32]) contains synthetic collagen textures. Although useful as a baseline, the synthetic dataset lacks the complexity and variability of real EHT samples. This mismatch must be acknowledged when interpreting generalization results.

Fair reporting. To avoid cherry-picking, we report metrics across multiple random seeds and include both successes and failures. Where preprocessing or model choices did not yield improvements, these are documented transparently to prevent bias.

Mitigations. Potential risks can be reduced by:

- Including human expert evaluation alongside automated metrics.
- Training with diverse datasets to reduce overfitting to synthetic or lab-specific conditions.
- Open-sourcing code, configurations, and (where possible) data to enable independent verification.

In summary, while deep learning offers powerful tools for biomedical imaging, reproducibility and ethical safeguards are essential for ensuring that results are both trustworthy and responsibly applied.

References

[1] Kaja Breckwoldt et al. "Differentiation of cardiomyocytes and generation of human engineered heart tissue". In: *Nature protocols* 12.6 (2017), pp. 1177–1197.

[2] Thomas Eschenhagen et al. "Physiological aspects of cardiac tissue engineering". In: *American Journal of Physiology-Heart and Circulatory Physiology* 303.2 (2012), H133–H143.

[3] Malte Tiburcy et al. "Defined engineered human myocardium with advanced maturation for applications in heart failure modeling and repair". In: *Circulation* 135.19 (2017), pp. 1832–1847.

[4] Daniel D. Streeter et al. "Fiber orientation in the canine left ventricle during diastole and systole". In: *Circulation Research* 24.3 (1969), pp. 339–347. DOI: 10.1161/01.RES.24.3.339.

[5] Peter A. Helm et al. "Ex vivo 3D diffusion tensor imaging and quantification of cardiac laminar structure". In: *Magnetic Resonance in Medicine* 54.4 (2005), pp. 850–859. DOI: 10.1002/mrm.20622.

[6] Warren R. Zipfel, Rebecca M. Williams, and Watt W. Webb. "Nonlinear optical microscopy: multiphoton and second harmonic generation imaging of biological tissues". In: *Journal of Investigative Dermatology* 116.5 (2001), pp. 838–845. DOI: 10.1046/j.1523-1747.2001.01377.x.

[7] Ph.D. Maria Cohut. *How dogs contribute to your health and happiness — medicalnewstoday.com*. https://www.medicalnewstoday.com/articles/322868. [Accessed 20-09-2025].

[8] *2.900+ Dog Far Away Stockfoto&apos;s, afbeeldingen en royalty-free beelden - iStock — istockphoto.com*. https://www.istockphoto.com/nl/fotos/dog-far-away. [Accessed 20-09-2025].

[9] LifeWithDogsAndCats. *Dog sitting in fog far away - Life with Dogs and Cats — lifewithdogsandcats.com*. https://lifewithdogsandcats.com/life-with-dogs-and-cats/my-dog-doesnt-like-to-get-his-feet-wet/attachment/jasper-fog-island-back-yard-distant-logo/. [Accessed 20-09-2025].

[10] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *ICML*. 2010.

[11] Shruti Jadon. *Introduction to Different Activation Functions for Deep Learning — shrutijadon*. https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092. [Accessed 06-10-2025].

[12] W3Schools. "AI Perceptrons". In: *W3Schools.com* (2025). Accessed: 15-09-2025. URL: https://www.w3schools.com/ai/ai_perceptrons.asp.

[13] Nadia Nedjah, Igor Santos, and Luiza Mourelle. "Sentiment analysis using convolutional neural network via word embeddings". In: *Evolutionary Intelligence* 15 (Apr. 2019). DOI: 10.1007/s12065-019-00227-4.

[14] UpGrad. "CNN Architecture: 5 Layers Explained Simply". In: *UpGrad Blog* (2025). Accessed: 15-09-2025. URL: https://www.upgrad.com/blog/basic-cnn-architecture/.

[15] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *ICML*. 2015.

[16] Matthew D Zeiler and Rob Fergus. *Visualizing and Understanding Convolutional Networks*. 2013. arXiv: 1311.2901 [cs.CV]. URL: https://arxiv.org/abs/1311.2901.

[17] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *ICLR* (2015).

[18] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[20] Jorge Nocedal and Stephen J Wright. *Numerical Optimization*. 2nd ed. Springer, 2006.

[21] Léon Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. In: *Proceedings of COMPSTAT’2010*. Springer, 2010, pp. 177–186.

[22] Lutz Prechelt. “Early stopping—but when?” In: *Neural Networks: Tricks of the Trade* (1998), pp. 55–69.

[23] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[24] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[25] *Deep Learning Computer Vision | Tensorflow | Image Classification Using deep learning Convolutional Neural Network | Google Cloud Run deployement — linkedin.com*. https://www.linkedin.com/pulse/deep-learning-computer-vision-tensorflow-image-using-run-shunmugaraj/. [Accessed 16-09-2025].

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[27] Liang Zou et al. “Spectral–Spatial Exploration for Hyperspectral Image Classification via the Fusion of Fully Convolutional Networks”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* PP (Feb. 2020), pp. 1–1. DOI: 10.1109/JSTARS.2020.2968179.

[28] Ozan Oktay et al. “Attention u-net: Learning where to look for the pancreas”. In: *arXiv preprint arXiv:1804.03999* (2018).

[29] Zongwei Zhou et al. *UNet++: A Nested U-Net Architecture for Medical Image Segmentation*. 2018. arXiv: 1807.10165 [cs.CV]. URL: https://arxiv.org/abs/1807.10165.

[30] Yunjiao Deng et al. “ELU-Net: An Efficient and Lightweight U-Net for Medical Image Segmentation”. In: *IEEE Access* 10 (Jan. 2022), pp. 1–1. DOI: 10.1109/ACCESS.2022.3163711.

[31] Jia Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[32] Hyojoon Park et al. “Collagen fiber centerline tracking in fibrotic tissue via deep neural networks with variational autoencoder-based synthetic training data generation”. In: *Medical image analysis* 90 (2023), p. 102961.

Part II

Scientific Article

Untangling the Heart: Automated Fiber Segmentation and Structural Metrics via Deep Learning

Arjun Vilakathara (4995074)

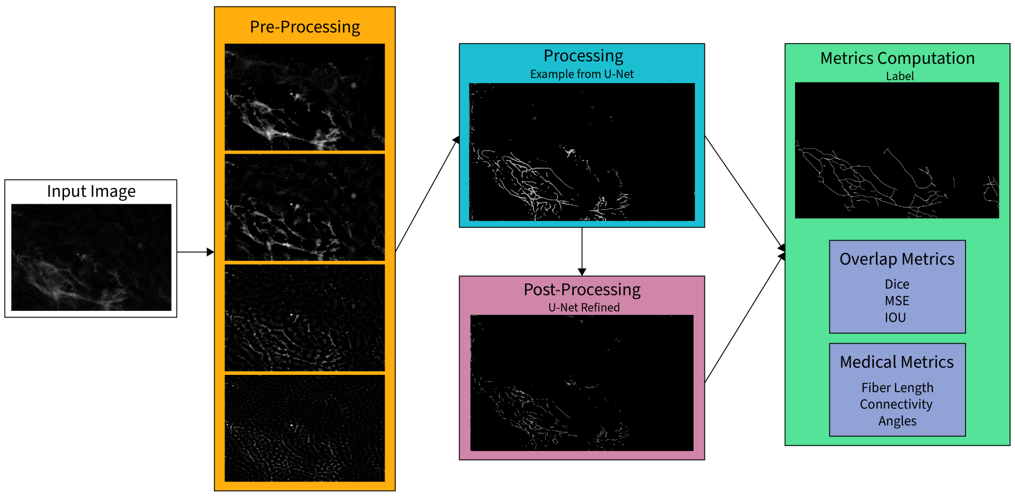


Figure 1: Overview of the proposed pipeline for automated fiber segmentation and structural analysis of engineered heart tissues (EHTs). Starting from a noisy confocal input slice (left), the pipeline applies pre-processing (orange) via FFT-based bandpass filtering to enhance mid to high frequency fiber structures. The enhanced slices are passed into segmentation models (blue), illustrated here with the U-Net model's output, and optionally refined through a secondary post-processing step (maroon). The resulting masks are compared against ground-truth annotations in the metrics stage (green), where both pixel-level overlap metrics (Dice, MSE, IoU) and biologically meaningful fiber-level metrics (length, connectivity, orientation) are computed. Together, this workflow bridges raw confocal imaging with quantitative, reproducible analysis of fiber architecture in EHTs , which can be used in downstream tasks such as evaluating muscle strength.

Abstract

Engineered heart tissues (EHTs) provide a promising platform for modeling cardiac physiology, but their dense and heterogeneous fiber organization makes quantitative analysis highly challenging. This thesis presents an automated pipeline for fiber segmentation and structural analysis of confocal EHT images. The framework integrates frequency based preprocessing using FFT bandpass filtering, state of the art deep learning segmentation models (U-Net, Attention U-Net, and U-Net++), and post-processing refinement through a secondary U-Net. Evaluation was conducted on a synthetic labeled dataset and on real EHT slices with sparse annotations. The results highlight clear trade-offs between model architectures. U-Net produced the most complete and connected fibers but introduced substantial hallucinations. Attention U-Net generated clean outputs but with fragmented fibers, and U-Net++ achieved a balance by capturing directionality and coherence with reduced continuity. Refinement net-

works were effective at reducing thickness and noise in some cases, but they often removed true fibers and fragmented long structures, providing limited overall benefit. Fiber level metrics and human inspection confirmed these findings, showing that orientation is captured reliably across models, while continuity and connectivity remain major challenges. Overall, the pipeline demonstrates the feasibility of automated structural analysis of EHTs and establishes a foundation for future work with improved datasets, advanced refinement strategies, and broader use of pretrained or structurally informed models.

1 Introduction

Advances in tissue engineering have enabled the creation of biomimetic constructs that replicate the structure and function of human organs. Among them, engineered heart tissues (EHT) have been shown as a powerful platform for studying cardiac physiology in a controlled setting. These lab-grown tissues are developed by culturing cardiomyocytes, often derived from human stem cells, within a 3D scaffold, resulting in a structure that exhibits contractile behavior similar to native heart muscle. EHTs are increasingly used as test-beds for drug screening, disease modeling, and regenerative strategies, offering a reproducible and physiologically relevant alternative to animal models [17].

A major advantage of EHTs lies in their potential for integration with computational tools. Due to these tissues being created and matured under well-defined conditions, they can be imaged and analyzed systematically. High-resolution imaging techniques such as confocal microscopy provide detailed views of the tissue architecture, capturing features like fiber orientation, density, and alignment. This allows for quantitative analysis and experimentation, which in itself presents a significant computation challenge. The resulting images are typically dense, noisy, and complex, making manual analysis both time-consuming and inconsistent [6] [11].

In this thesis, we develop a machine learning–based pipeline for analyzing the internal fiber architecture of EHTs from confocal microscopy data. Specifically, we aim to segment individual fibers and extract structural metrics that can be used to construct a graph-based model of the tissue. This model supports downstream analysis in a separate biomechanical simulation project, which studies how fiber orientation influences tissue behavior under mechanical load.

To achieve this, we explore the use of deep learning segmentation models, along with tailored preprocessing and post-processing techniques designed to enhance fiber visibility and separability. The resulting framework enables automated, reproducible analysis of EHT structure, contributing to the broader goal of bridging experimental tissue engineering with computational modeling.

We evaluate three deep learning segmentation architectures: U-Net, Attention U-Net, and U-Net++ (with and without pretrained backbones), on engineered heart tissue images. Models were trained on a synthetic collagen dataset and tested both on this labeled dataset and on real EHT confocal slices with limited manual annotations. We further explored FFT-based frequency preprocessing, patch-based inference, and a U-Net–based refiner network for post-processing. The results reveal clear trade-offs between architectures: U-Net achieves the highest recall and produces visually complete fibers but introduces many hallucinations, while Attention U-Net yields clean but fragmented outputs. U-Net++ provides a more balanced compromise, with the pretrained version offering the best overall Dice score on the labeled dataset. Post-processing refiners were generally ineffective, often reducing recall rather than improving continuity. Overall, the pipeline demonstrates that meaningful structural metrics can be extracted from challenging EHT data, but that model choice strongly influences the balance between completeness and precision.

2 Related Works and Motivation

Automated detection and segmentation of fibrous networks in biological tissues has been studied across multiple domains, ranging from engineered microtissues to pathological human samples. In this section, we group related works into three domains: (i) imaging and datasets for engineered tissues, (ii) segmentation methods for biomedical microscopy images, and (iii) fiber-specific segmentation and analysis pipelines. This is to clarify the context of our work and highlights the specific gap addressed by this thesis.

2.1 Imaging and Datasets for Engineered Tissues

Engineered heart tissues (EHTs) are increasingly used for drug screening and disease modeling, requiring quantitative analysis of their internal architecture. Foundational studies such as Tiburcy et al. [17] and Greiner et al. [6] highlight how confocal imaging enables detailed views of cardiac tissue organization, including fiber orientation and remodeling. However, annotated datasets for EHTs remain scarce.

To address this, synthetic datasets have been introduced. Notably, the Synthetic MicroBundle dataset by Kobeissi et al. [10] simulates fibrous microtissues and provides a framework for segmentation benchmarking. While powerful, the synthetic fibers are simpler and more homogeneous than the dense, heterogeneous networks found in real EHTs. Similarly, Park et al. [14] proposed a GAN-based pipeline to generate fibrotic-like collagen datasets for training segmentation models. This synthetic dataset, which we use as our Labeled dataset, enables training but does not fully capture the density and complexity of EHTs. The contrast between realistic but under-annotated EHT data and richly annotated but synthetic datasets motivates the dual-dataset strategy adopted in this thesis.

2.2 Segmentation Methods in Biomedical Imaging

Segmentation of biomedical microscopy images has been commonly done through convolutional encoder–decoder architectures. U-Net, introduced by Ronneberger et al. [15], established the baseline for biomedical segmentation. Extensions such as Attention U-Net [13] added gating modules to improve focus on relevant features, while U-Net++ [21] introduced nested skip connections for better multi-scale feature integration. Beyond these, 3D extensions such as 3D U-Net [5] and V-Net [12] adapt the encoder–decoder design to volumetric data, directly relevant for confocal stacks of EHTs.

Other approaches have emphasized improved training objectives and refinement strategies. For example, boundary-aware losses [8] and residual refinement networks [4] were developed to mitigate structural fragmentation in medical segmentation tasks. These studies demonstrate that both architectural choices and loss design strongly influence the ability of networks to recover fine structures such as fibers.

2.3 Fiber Segmentation and Analysis

Several methods specifically target collagen or fibrous structures. Xu et al. [19] presented one of the earliest quantitative frameworks for analyzing fiber orientation and morphology in SHG microscopy. Bredfeldt et al. [3] developed CT-FIRE, which integrates curvelet-based denoising with fiber-tracking, providing fiber-level metrics such as length and curvature. More recently, Stein et al. [16] demonstrated deep learning for collagen fiber segmentation in lung tissue SHG images, showing the feasibility of learning-based methods for dense fibrous networks. Kobat et al. [9] further introduced a deep learning approach for collagen fiber centerline tracking, combining synthetic SHG-like data with supervised training. These works highlight both the promise and the difficulty of extracting reliable fiber-level metrics from noisy, high-density microscopy data.

In contrast to these studies, our focus is on dense and heterogeneous EHT images captured by confocal microscopy. We systematically evaluate three architectures (U-Net, Attention U-Net, and U-Net++), combined with FFT-based preprocessing and U-Net–based refiners, to extract biologically meaningful metrics such as fiber length, orientation, and connectivity. Unlike prior fiber segmentation pipelines, which often focus on sparse or homogeneous networks, our pipeline explicitly addresses the challenges of EHT imaging and emphasizes metric-level analysis alongside pixel-level evaluation. This end-to-end approach bridges imaging, segmentation, and structural analysis, providing a proof of concept for integrating computational pipelines with engineered tissue research.

3 Dataset

This section describes the two datasets and any processing needed on them to be used throughout this thesis for training and evaluation.

3.1 Delft EHT Dataset

The primary dataset comprises of three unlabeled volumetric (3D) confocal-microscopy images of engineered heart tissues (EHTs), originally acquired and preprocessed in Imaris. Because Imaris's processing and image conversion code are not accessible for Python, we use a third-party library, imaris file reader [1], to extract each volume as a stack of TIFF slices (see Figure 2 for one such slice). These raw slices contain a uniform background noise floor, which we estimate by sampling the feature-free regions at the very edges of the image and then subtracting that value from every pixel. After noise subtraction, each slice is normalized to the [0,255] range (Figure 3). This denoising and normalization pipeline is applied independently to every layer in all three volumes.

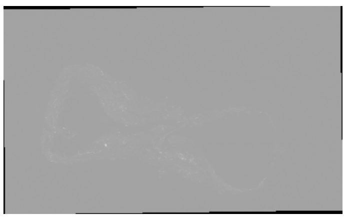


Figure 2: Raw slice from the Delft EHT dataset showing the uniform background noise floor. This illustrates the preprocessing challenge: noise dominates the image and must be subtracted before segmentation.

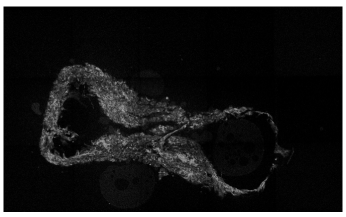


Figure 3: The same Delft EHT slice after noise subtraction and normalization to [0,255]. Fiber structures become visible, providing the true input for downstream segmentation.

Zooming into a small crop of a normalized slice (Figure 4) may appear nearly empty, in reality, there are fiber structures present which become evident when the same crop is displayed on a color calibrated BenQ EL2870U monitor as seen in Figure 5. For visualization purposes in this thesis, we show the image in figure 3 with boosted pixel values, which look like figure 6. All quantitative segmentation and metric computations are performed on the raw, denoised volumes without any additional intensity enhancement.

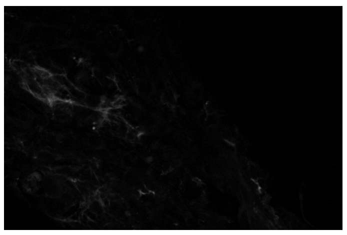


Figure 4: Zoomed crop of Figure 3 (Delft EHT dataset). Although the crop appears nearly empty at first glance, faint fiber structures are present, highlighting the difficulty of detecting subtle fibers.

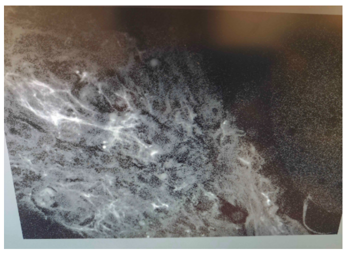


Figure 5: The same crop from Figure 4 displayed on a color-calibrated BenQ monitor. Fine fiber details become visible, showing the true detail of the slice.

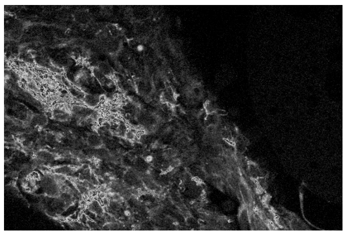


Figure 6: The crop from Figure 4 with boosted pixel intensities (visualization only). This artificial enhancement makes fiber structures clearer for the reader, though all quantitative analysis uses the normalized but unboosted images.

To provide ground-truth reference values for evaluation, we manually annotated a small subset of slices from this dataset. Given that these annotations were produced by a novice labeler without extensive domain experience, they are not intended as a gold-standard segmentation. Rather, they serve as a rough reference to evaluate the relative performance of our methods.

An example of the annotation of figure 3 can be seen in figure 7.

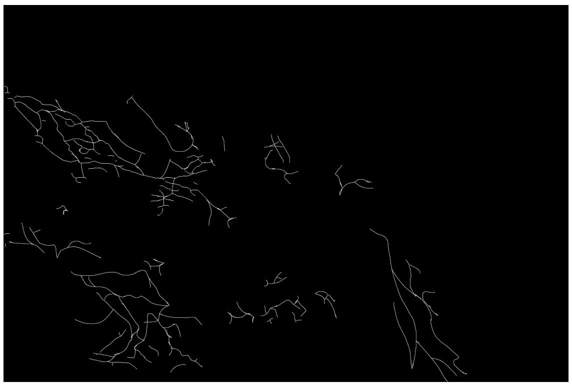


Figure 7: Delft EHT dataset: manual annotation (ground truth) for Figure 3. Provides the reference mask for evaluation, noting conservative labeling by a novice annotator.

3.2 Synthetic Labeled Dataset

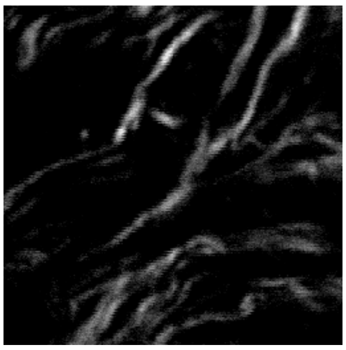


Figure 8: Labeled synthetic dataset: representative input image from Park et al. data. Shows the simpler, cleaner appearance used for training/evaluation of baseline models.

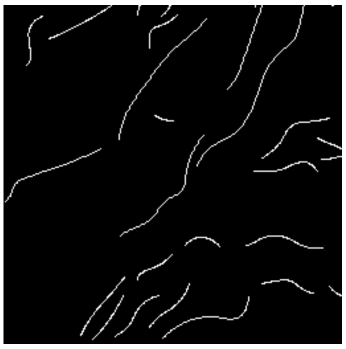


Figure 9: Labeled synthetic dataset: corresponding ground-truth mask for Figure 8, used for supervised training and evaluation.

To validate and train our methods, we need to use a publicly available dataset. The work in [14] describes how they used a GAN to generate training data, similar to fibrotic tissue, given labels. They provide both training data of 1200 images and test data of 300 images, which we utilize to train and evaluate the methods that will be discussed later. An example of the image and label can be seen in figure 8 and 9 respectively.

4 Pipeline

This section will go over the entire pipeline used to process the EHT images. We first discuss the Pre-processing in Section 4.1, followed by the different models used to process the images in Section 4.2, followed by the Post-Processing steps in Section 4.3, which describe the models used to refine the raw outputs, and conclude with how the metrics are computed in Section 4.4.

The figure in image 10 shows a visual representation of how the image is transformed into the output label and metrics.

4.1 Pre-Processing

The raw images contain features of varying frequencies. As seen in Figure 5, the fibers that appear in bright white are surrounded by a low-frequency background of grey features. Similarly, some features toward the top right resemble salt-and-pepper noise: dots packed closely together. These non-fiber components are not relevant to our segmentation task, so removing them improves downstream processing. Because we are specifically targeting

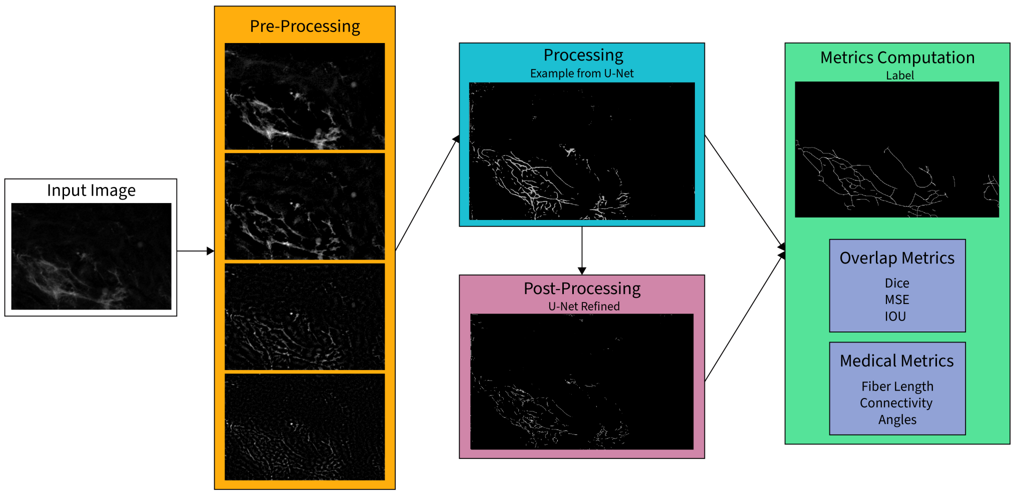


Figure 10: Ooverview for Section 4: FFT bandpass pre-processing (orange), model inference with U-Net / Attention U-Net / U-Net++ (blue), optional U-Net-based refinement (maroon), and metric computation from skeletonized masks (green) covering Dice/MSE and fiber length, connectivity, and orientation. Repeated here from Figure 1 for convenience.

mid-frequency fiber boundaries, we safely discard both lower- and higher-frequency content. Therefore, we explore both Highpass–Lowpass filtering and FFT-based bandpass techniques.

4.1.1 Highpass–Lowpass Filtering

Highpass–Lowpass filtering works by convolving the image with kernels that attenuate undesired frequency bands. A lowpass filter smooths out high-frequency noise (e.g., salt-and-pepper), while a highpass filter removes low-frequency background. Combining these operations results in a bandpass effect that preserves the frequencies of the fibers.

4.1.2 FFT-Based Bandpass

Using the Fourier transform to pre-process the image allows for very precise frequency selection. Given that the fiber details we are interested are lie between a specific frequency range, we can use this transform to precisely remove the excess frequencies. Below, we describe how the FFT functions.

The 2D Fourier transform of an image $f(x, y)$ of size $M \times N$ is:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \exp\bigl[-2\pi i\,(ux/M+vy/N)\bigr].$$

To isolate mid-range frequencies, we define a bandpass mask based on the frequencies we wish to preserve $R_{\text{low}}, R_{\text{high}}$:

$$H(u,v) = \begin{cases} 1, & R_{\text{low}} \le \sqrt{u^2+v^2} \le R_{\text{high}}, \\ 0, & \text{otherwise}, \end{cases}$$

and apply it:

$$\widetilde{F}(u,v) \;=\; F(u,v)\,H(u,v).$$

The filtered image $\tilde{f}(x, y)$ is recovered via the inverse FFT:

$$\tilde{f}(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \widetilde{F}(u,v) \exp\bigl[2\pi i\,(ux/M+vy/N)\bigr].$$

A Visual description of this process is described in detail in the supplementary section of the paper at Section 2.3.2.

4.2 Processing

For segmentation, we selected three convolutional encoder–decoder architectures commonly applied in biomedical imaging: U-Net, Attention U-Net, and U-Net++. These models were chosen because they share a comparable encoder–decoder backbone while having some architectural variations that enable evaluation of their relative strengths and weaknesses for engineered EHT images.

To ensure fair comparison, all models were trained under identical conditions described in section 5 Training.

This setup ensures that performance differences can be attributed primarily to model architecture, rather than variations in training parameters.

4.2.1 U-Net

We implemented a standard U-Net based on the architecture introduced by Ronneberger et al. [15], consisting of a symmetric encoder–decoder structure connected through skip connections. Our implementation begins with 64 filters in the first convolutional block, doubling the number of filters at each downsampling stage. The decoder employs bilinear upsampling to reconstruct the segmentation map. This model serves as the baseline against which the other architectures are compared.

For this thesis, we build upon the U-Net variant pretrained and evaluated by Park et al. [14], originally developed for collagen fiber detection in SHG microscopy images. Their pipeline has demonstrated robustness in segmenting fibrous biological textures, making it a suitable foundation for adaptation to engineered heart tissue segmentation. Since their publicly available implementation and pretrained weights share the same architectural design as our U-Net, we use their model trained on the secondary dataset described in Section 3.2.

4.2.2 Attention U-Net

Attention U-Net extends the baseline by introducing attention gates at skip connections, allowing the network to focus on relevant fiber structures and suppress irrelevant background features. The implementation is based on Oktay et al. [13], with attention blocks inserted before concatenation in the decoder. Aside from these gates, the architecture and training parameters remain aligned with the baseline U-Net for comparability.

4.2.3 U-Net++

For U-Net++, we used the segmentation-models-pytorch implementation of U-Net++. The library also allows the option to use pretrained encoders, which allow us to compare the results:

U-Net++ (base): configured with no pretrained encoder weights. This ensures the model learns entirely from the fiber datasets and allows a fairer comparison with our custom U-Net and Attention U-Net implementations.

U-Net++ (pretrained): configured with a ResNet34 backbone initialized on ImageNet. This version leverages transfer learning, which allows us to see if and how the performance improves.

Both models used the same setup as in Sections 5.1–5.2 (input size, output, optimizer, loss function, epochs), ensuring comparability with U-Net and Attention U-Net.

While U-Net++ introduces nested skip connections and dense pathways that distinguish it from standard U-Net [21], keeping the training conditions identical allows us to attribute performance differences either to the architecture and/or to the presence of pretrained features.

4.3 Post-Processing

This section describes the post-processing stage of the pipeline. When comparing the processed output images to their ground truth labels, a few common errors are observed (Figure 11 showcases these errors):

1. Disconnected Lines: Continuous fibers in the labels may appear broken in the predictions.
2. False Positives: False positive fibers, and often noise such as dots, appear where in reality the region contains no features.
3. Disconnected Intersections: Points where fibers intersect in the label may appear as separate, unconnected segments in the predictions.

These artifacts would affect the metric computations. The goal of this step is to reduce the errors. We first present a rule-based algorithm that uses manually tuned thresholds in section 4.3.1, followed by a learned refinement strategy using the Refiner U-Net in section 4.3.2.

The Refiner U-Net used here follows the idea of image-to-image translation, where a network is trained to map one structured representa-

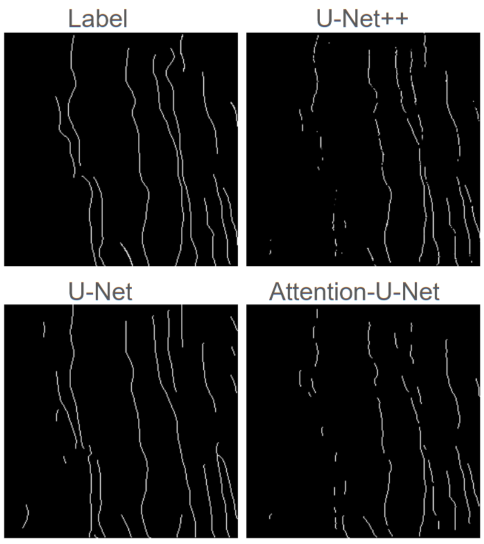


Figure 11: Prediction errors vs. ground truth (labeled dataset examples), highlighting broken fibers, false positives, and disconnected intersections.

tion into another. This concept is used by conditional GAN approaches such as Pix2Pix [7], which learn to transform edge maps or semantic layouts into realistic images. Our refiner is similar in concept, as it also maps one image-like representation (a segmentation mask) to a more refined version, but differs in two key aspects: (i) it is trained only with direct supervision against ground-truth masks rather than adversarial objectives, and (ii) its goal is not photorealism but structural correction (removing hallucinations, reconnecting fragmented fibers). Related approaches in biomedical imaging have also explored U-Net–based refinement of coarse predictions [20, 18], showing that such post-processing networks can improve local structural quality.

4.3.1 Fiber clustering through manual input

To analyze the fiber structures, we first skeletonize the image using the skeletonize function from Scikit-Image [2]. This reduces the fibers to one-pixel-wide lines, making the structure simpler to process with neighborhood-based algorithms.

After skeletonization, we use connected component analysis to label each group of connected pixels. This step helps in identifying continuous fibers in the image. Figure 12 shows the output from this step where each detected fiber is given a unique color.

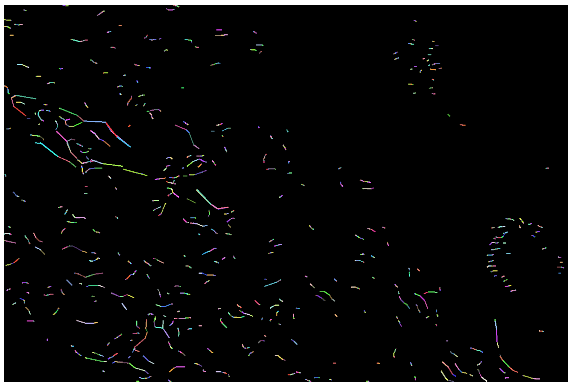


Figure 12: EHT dataset: skeletonized predictions with connected components color-coded.

Fibers are then connected by checking if any pixel from one fiber is within a defined proximity threshold of any pixel from another fiber. In which case, the fibers are then merged. Figure 13 shows how the fibers are reconnected from the result in Figure 12.

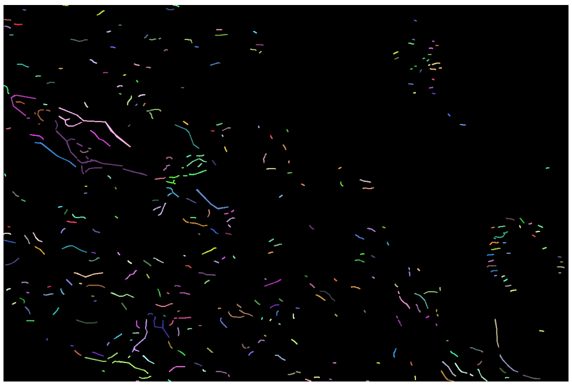


Figure 13: EHT dataset: proximity-based merging of connected components, reconnecting near-miss segments into more continuous fibers. Used to compute length and orientation metrics.

Although effective, this method is limited by the user-defined threshold that determines when two fibers are considered connected. If the threshold is set too high, it could result in the entire image being considered as a single fiber. While this approach is useful, it suffers from significant computational inefficiency. The time complexity is poor because the algorithm checks every pixel against every other pixel across all fibers, making the process computationally expensive, especially for large images. While this method provides a possible solution, a more elegant approach would be preferable to reduce computational time.

4.3.2 Refiner U-Net

The Refiner U-Net method is another approach to improve fiber connectivity. In this method, we use the outputs of the existing models (inferred images) as inputs to a Refiner U-Net. The idea is that this refiner network learns to refine the model's output towards the ground truth labels, improving the accuracy of fibers in the outputs.

The Refiner U-Net takes the original model output and the same ground truth labels used previously, training to minimize the difference between the predicted fibers and the true fiber structure. The network learns to correct any disconnects in the output, filling in gaps and reducing False positives. The U-Net architecture used in this refinement step is identical to the one described in section 4.2.1, which has been proven effective for segmentation tasks. By fine-tuning the model output with this additional refinement step, we can achieve more accurate results with lower computational intensity than the manual thresholding method.

Figure 14, which shows the result of the refining pipeline on the training images, demonstrates that the method works. However, this image is as good as it is since it is part of the data used to train the model, and refinements on real data may not be this good.

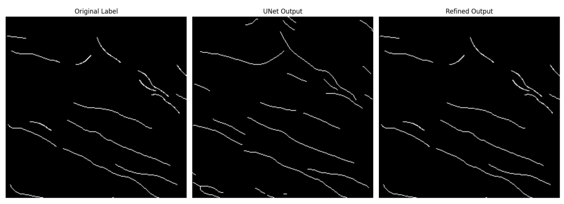


Figure 14: Labeled dataset: example of base prediction and refined output alongside ground truth, illustrating intended refiner behavior on training images.

4.4 Metric Computation

This section describes the metric computation pipeline, which calculates key medically relevant properties of the segmented fibers, including length, angle, and connectivity. The process begins by skeletonizing the segmented image, as described in Section 4.3.1. Once the image is skeletonized, the following metrics are computed.

4.4.1 Length Calculation

Fiber lengths are computed by identifying each individual fiber segment in the skeletonized image using connected component labeling. The length of each fiber is calculated by summing the number of pixels in that component. The number of pixels associated with each label directly corresponds to the fiber length. The real-world length is then computed by multiplying the pixel-based length by the physical size of a pixel in real-world units (micrometers per pixel).

4.4.2 Angle Calculation

Fiber orientation is estimated by calculating the angle of each individual fiber segment based on its endpoints. For each connected component identified in the skeletonized image, the endpoints are determined, and a straight line is fit between them. The orientation angle θ is then computed using the arctangent of the slope:

$$\theta = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right)$$

where (x_1, y_1) and (x_2, y_2) are the coordinates of the two endpoints of the fiber segment. The resulting angle θ is the direction of the fiber relative to the horizontal axis. This method provides an approximate but effective measure of fiber alignment across the image.

4.4.3 Connectivity Calculation

Connectivity is evaluated by identifying points in the skeletonized image where multiple fibers join or intersect. This is done by checking the number of neighboring pixels around each pixel in the skeleton.

For each pixel, we check how many of the surrounding pixels (in all directions i.e. the 3x3 neighborhood of the pixel) are also part of the fiber. If a pixel is connected to more than two other fiber pixels, it is considered a branching or intersection point. These points indicate where fibers are connected.

To avoid overcounting due to closely clustered intersections, a small distance threshold is applied to group nearby intersection points together. The total number of such grouped intersections gives a measure of the overall connectivity in the image.

Figure 15 shows an example where the detected intersection points are highlighted in red.

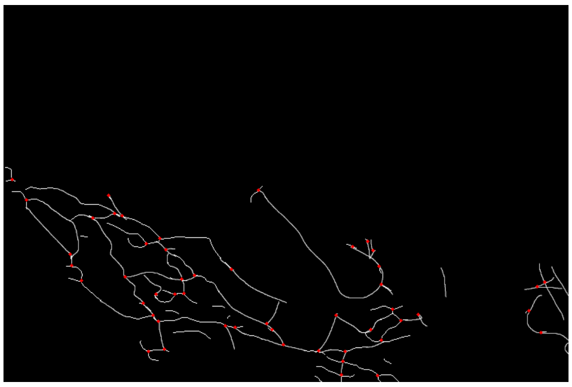


Figure 15: EHT dataset: detected intersection points highlighted in red on a skeletonized mask, showing how connectivity is computed.

Interpretation of Metrics. It is important to note that the computed fiber metrics (length, orientation, and connectivity) represent numerical approximations rather than true physical measurements. They are derived from the pixel geometry of the binary segmentation masks and do not capture the exact structure of the underlying fibers. However, because the same metric extraction algorithm is applied consistently to both the ground-truth labels and the model predictions, the resulting values remain comparable across experiments. Thus, while these metrics do not reflect absolute real-world quantities, they provide a reliable and reproducible result for our purposes.

5 Training

This section describes the training process for the models in this thesis. In Section 5.1, we outline the training setup, including data preparation and hardware configuration. In Section 5.2, we present the loss functions used to optimize the models. Finally, in Section 5.3, we describe the saving strategy adopted to preserve checkpoints and select the best-performing models.

5.1 Training Setup

The training dataset consisted of paired images and binary masks representing the centerlines of collagen fibers. All images and masks were resized to 256×256 pixels and normalized using standard PyTorch transformations. A random 80/20 train-validation split was applied.

Each model was trained using the Adam optimizer with a learning rate of 1×10^{-4} for 1200 epochs. A batch size of 8 was used throughout. The training was conducted on the NVIDIA Tesla P100 GPU.

5.2 Loss Function

Given the nature of the segmentation task: detecting thin, sparse collagen fibers. We use a composite loss consisting of Binary Cross-Entropy (BCE) and Dice loss:

$$\mathcal{L}_{\text{total}} = \text{BCE} + \text{DiceLoss}$$

Binary Cross-Entropy (BCE) Loss. BCE loss evaluates pixel-wise differences between the predicted and ground truth masks:

$$\mathcal{L}_{\text{BCE}}(p,t) = -\left[t\log(p) + (1-t)\log(1-p)\right]$$

where p is the predicted probability, and t is the ground truth label (0 or 1).

Dice Loss. Dice loss measures the overlap between the predicted mask and the ground truth mask:

$$\mathcal{L}_{\text{Dice}}(p,t) = 1 - \frac{2 \cdot |p \cap t| + \epsilon}{|p| + |t| + \epsilon}$$

$|p \cap t|$ denotes the intersection between prediction and target, and a very small value ϵ is added to avoid division by zero.

Combined: The combined loss leverages the strengths of both methods: BCE contributes to pixel-wise correctness, while Dice loss promotes structural coherence and robustness to imbalance. The combination provides a better way to evaluate the error compared to just using either.

5.3 Saving Strategy

Each model was trained for a total of 1200 epochs, with checkpoints saved at epoch 500, epoch 1000, and at the end of training. In addition, we maintained a separate checkpoint for the *best performing model*, defined as the epoch with the lowest validation loss. This model was saved independently of the fixed-epoch checkpoints.

6 Experiments

The experiments in this thesis were designed to systematically evaluate how preprocessing, model choice, and refinement strategies affect segmentation quality and structural metric extraction in EHT images. To maintain clarity, the experiments are grouped into six categories, which correspond directly to the subsections in Section 7 (Results and Discussion).

The overall pipeline (Section 4) is followed in all cases: each image was preprocessed, segmented with a trained model, optionally refined, and then converted into both pixel-level and fiber-level metrics.

6.1 Inference Setup

Since all models were trained on 256×256 pixel images from the secondary dataset, inference on the larger slices of the primary dataset required a patch-based approach. Each confocal slice was divided into 256×256 patches, predictions were made independently for each patch, and the results were stitched together. This ensured compatibility with the training setup while allowing full-resolution evaluation.

6.2 Low-Cut Preprocessing Experiments

The effect of frequency-domain preprocessing was evaluated by varying the low-cut parameter of the FFT-based bandpass filter. In the Fourier domain, the low-cut specifies the minimum retained spatial frequency, expressed as the radius of the central region that is suppressed. Lower values preserve large-scale intensity variations, whereas higher values progressively remove background structure and emphasize finer detail.

For the labeled dataset, low-cut values from 0 to 10 were tested. Here, low-cut 0 retains all frequencies, while low-cut 10 suppresses all content within the central radius of 10 pixels in Fourier space. For the EHT dataset, which has larger image dimensions and correspondingly higher frequency resolution, the sweep was extended to 0–20 to cover an equivalent range of scales.

Each architecture was evaluated across these low-cut values, with results compared in terms of Dice scores and confusion matrix components. This experiment was used to determine the optimal low-cut setting for subsequent evaluations and to assess how frequency suppression interacts with model performance.

6.3 Baseline Models

We then evaluated the four baseline segmentation models described in Section 4.2:

- U-Net (baseline)
- U-Net++ (base, no pretrained encoder)
- U-Net++ (pretrained, ResNet34 backbone)
- Attention U-Net

For each model, predictions were generated on both the labeled test dataset and the EHT dataset. These baseline outputs provided the foundation for all subsequent refinements and comparisons.

6.4 Refinement with Refiner U-Net

To investigate whether structural inconsistencies could be corrected as a post-processing step, we trained a Refiner U-Net for each model. The refiner took base model outputs as input and ground-truth masks as targets, with the goal of improving fiber continuity and suppressing hallucinations. Refined predictions were then compared to baseline outputs across all metrics.

6.5 Threshold Sweeps

Because refiner outputs are continuous probability maps, we tested multiple binarization thresholds (0.3, 0.5, 0.7, 0.9) to explore the precision–recall trade-off. A threshold of 0.5 was used as the default, but additional sweeps allowed us to assess whether alternative thresholds could provide more favorable trade-offs.

6.6 Fiber-Level Metric Evaluation

Beyond pixel-level metrics, we computed biologically meaningful fiber-level properties:

- Distribution of and mean fiber length (pixels)

- Distribution of and mean fiber orientation (degrees)
- Distribution of and connectivity score (intersection count)

These were derived from skeletonized masks and aggregated across datasets, providing medically relevant structural context to complement Dice, MSE, and True Positive / False Postive statistics.

6.7 Human Inspection

Finally, we complemented quantitative analysis with a qualitative inspection of all model and refiner outputs. Outputs from all models were visually compared against ground truth labels, focusing on perceptual continuity, hallucinations, and the overall overlap of predicted fiber structures with the ground truth.

7 Results and Discussion

This section presents results in the following order. We first analyze the impact of frequency pre-processing via the low-cut parameter, which determines how much low-frequency content is suppressed before inference. We begin with the labeled dataset to, then perform the same analysis on the EHT dataset to find the architecture-specific optimal lowcut boundary. Subsequent subsections (baseline models, refiner, threshold sweeps, and fiber-level metrics) will use these chosen low-cut settings to ensure fair and informative comparisons.

In order to interpret the quantitative results presented in this section, we first clarify the meaning of the basic evaluation terms used throughout. A true positive (TP) refers to a pixel that is correctly predicted as belonging to a fiber, while a false positive (FP) refers to a pixel that is predicted as a fiber but is not annotated as such in the ground truth. Conversely, false negatives (FN) are fiber pixels that are missed by the model, and true negatives (TN) are background pixels correctly identified as non-fiber. The balance between TP, FP, FN, and TN underlies the reported metrics such as Dice and mean squared error (MSE), and provides a more detailed picture of model behavior.

7.1 Effect of Low-Cut Frequency Preprocessing (Labeled Dataset)

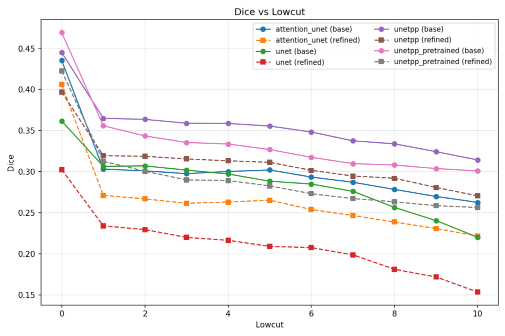


Figure 16: Labeled dataset: Dice vs. lowcut for all models. Dice declines with increasing lowcut.

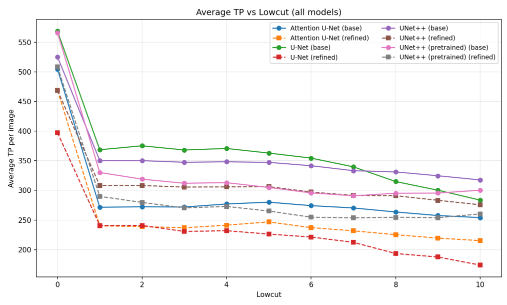


Figure 17: Labeled dataset: TP vs. lowcut. TP is highest at lowcut 0 and falls gradually up to ∼5, then drops steeply beyond 6 as useful fiber content is filtered out. **Takeaway:** modest lowcut (2–5) preserves recall while enabling cleaner outputs.

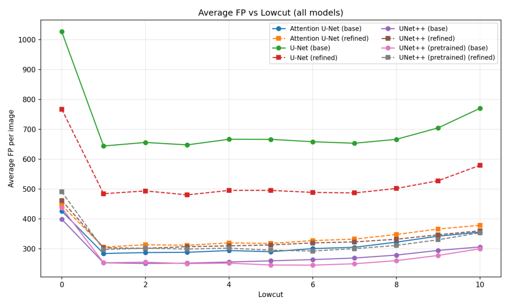


Figure 18: Labeled dataset: FP vs. lowcut. FP decreases sharply from 0 to ∼2, then flattens or even rises for higher lowcut as models begin to hallucinate missing structure. **Takeaway: lowcut 5** seems to be a good choice as it is relatively flat before, and FP increases after 6.

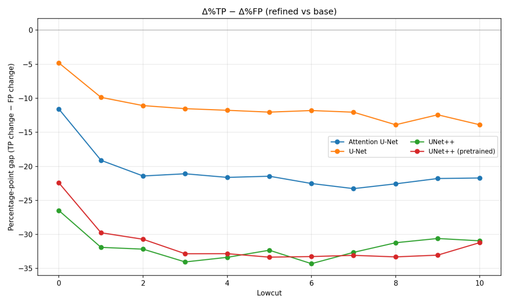


Figure 19: Labeled dataset: percentage-point gap between ΔTP and ΔFP (refined vs. base). Negative values indicate that refinement removes more TP than FP. This show sus that this gap increases as the lowcut increases, which means refiners remove even more TP's than FP's as lowcut increases.

Figure 16 shows that Dice scores decrease as the lowcut threshold increases. However, the underlying TP/FP curves (Figures 17-18) reveal a more informative trade-off. At lowcut 0, both true and false positives are the highest: recall (TP) is high, but hallucinations (FP) dominate. Introducing moderate filtering (lowcut 2–5) substantially reduces FP while TP declines only mildly, producing cleaner and more interpretable segmentations. Beyond lowcut 6, TP drops steeply and FP paradoxically begins to rise again, indicating that essential fiber content is being removed and the models begin to hallucinate structures to compensate.

This overall trend is complemented by the TP–FP gap in Figure 19, which compares refined outputs to their corresponding base models. Negative values here indicate that the refiner removes more TP than FP, illustrating that refinement often suppresses true fibers more aggressively than hallucinations. While this figure does not describe the direct effect of lowcut filtering, it helps to interpret how refinement interacts with the filtered outputs.

Taken together, these results suggest that lowcut 5 provides the most balanced trade-off on the labeled dataset: it suppresses hallucinations meaningfully without excessively reducing recall. Therefore, unless stated otherwise, all labeled-dataset comparisons that follow use lowcut 5 FFT preprocessing before all inferences.

7.2 Effect of Low-Cut Frequency on EHT Data

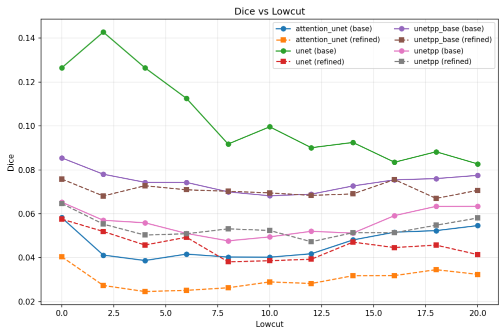


Figure 20: EHT dataset: Dice vs. lowcut. U-Net peaks at **lowcut 2**; Attention U-Net and both U-Net++ variants are flatter with gentle optima near **lowcut 8**. **Takeaway:** choose **2 for U-Net**, **8 for Attention/U-Net++**.

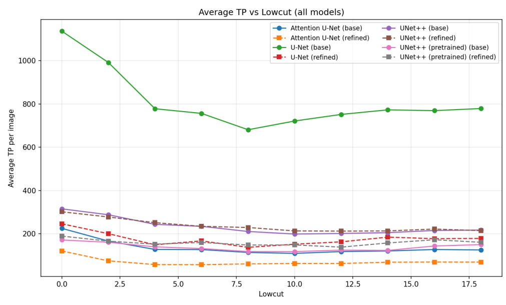


Figure 21: EHT dataset: TP vs. lowcut. U-Net maintains high TP at lowcut 2 but loses TP as lowcut rises; other models degrade more gently. **Takeaway:** less aggressive filtering benefits U-Net on EHT.

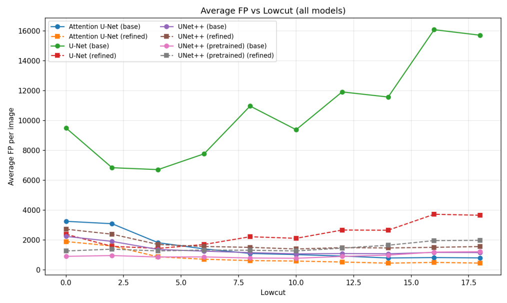


Figure 22: EHT dataset: FP vs. lowcut. U-Net's FP is minimized near lowcut 2, Attention/U-Net++ keep FP controlled up to ∼8 before trade-offs worsen. **Takeaway:** 2 for U-Net, 8 for Attention/U-Net++ balance FP with good TP.

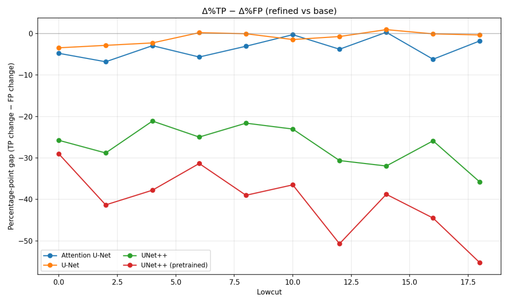


Figure 23: EHT dataset: percentage-point gap between ΔTP and ΔFP (refined vs. base). Negative values indicate that refinement removes more TP than FP.

Applying the same analysis to the EHT dataset yields a more architecture-specific picture (Figures 20–23). For the base U-Net, Dice peaks at lowcut 2 (Figure 20), which also coincides with one of its lowest FP values (Figure 22) while TP remains high (Figure 21). This indicates that U-Net benefits from a less aggressive filter that suppresses only the most obvious noise.

By contrast, Attention U-Net and both U-Net++ variants display flatter Dice curves with a gentle optimum around lowcut 8. At this setting, TP is not overly suppressed, FP remains low, and Dice is near the maximum for each model. The percentage-point gap analysis in Figure 23, which compares refined outputs against their base counterparts, further refines this view. Unlike the labeled dataset, where the gap values were initially small, the U-Net++ models show consistently high negative values, indicating that refinement removes substantially more TP than FP. In contrast, Attention U-Net and the base U-Net remain relatively flat and low in percentage change, meaning they lose only marginally more TP than FP regardless of lowcut. While this figure does not directly describe the effect of lowcut filtering, it highlights how refinement interacts with filtered outputs and why its impact differs between architectures.

Therefore, for the experiments below, U-Net is evaluated at lowcut 2, Attention U-Net at lowcut 8, and U-Net++ (base and pretrained) at lowcut 8. These settings reflect each architecture's most balanced trade-off between TP, FP, and Dice on the EHT dataset.

7.3 Baseline Model Performance

Unless otherwise specified, labeled-dataset results use lowcut 5 (Section 7.1); EHT results use the best-performing lowcut per model (Section 7.2), i.e. 2 for U-Net and 8 for Attention U-Net and U-Net++.

Before presenting baseline results, we briefly define the evaluation metrics used in this section. Precision measures the fraction of predicted fiber pixels that are correct:

$$\text{Precision} = \frac{TP}{TP + FP}.$$

Recall measures the fraction of ground-truth fiber pixels that are correctly detected:

$$\text{Recall} = \frac{TP}{TP + FN}.$$

High precision indicates that few hallucinations are produced, while high recall indicates that most true fibers are recovered. Together with Dice and MSE, these metrics provide a more complete picture of model behavior.

7.3.1 Baseline Model Performance on Labeled Data

Table 1 summarizes the average TP, FP, and FN values, along with derived precision and recall, for the labeled dataset.

U-Net achieves the highest recall (0.46), meaning it detects the most true fibers, but at the cost of very low precision (0.36) due to its high FP rate. Attention U-Net and both U-Net++ variants achieve substantially higher precision (0.54–0.57), reflecting their ability to suppress hallucinations, but at the cost of lower recall (0.40–0.45). Thus, U-Net prioritizes completeness, whereas U-Net++ and Attention U-Net prioritize precision. Between the U-Net++

Table 1: Labeled dataset: average TP, FP, FN, and derived precision/recall at lowcut 5. U-Net achieves the highest recall; U-Net++ (pretrained) is most balanced.

Model	TP	FP	FN	Precision	Recall
U-Net	578	1040	685	0.36	0.46
U-Net++ (base)	527	399	736	0.57	0.42
U-Net++ (pretrained)	568	437	695	0.57	0.45
Attention U-Net	509	432	754	0.54	0.40

variants, U-Net++ (pretrained) achieves the most balanced trade-off.

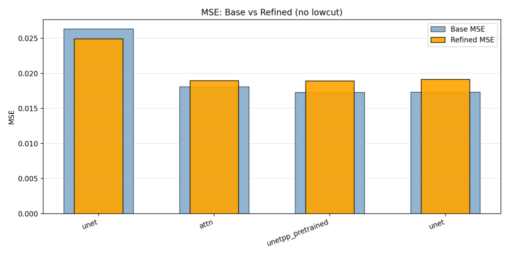


Figure 24: Labeled dataset: MSE, base vs. refined. Refinement reduces MSE for U-Net but worsens it for the other models. **Takeaway:** refiners do not universally improve pixel similarity.

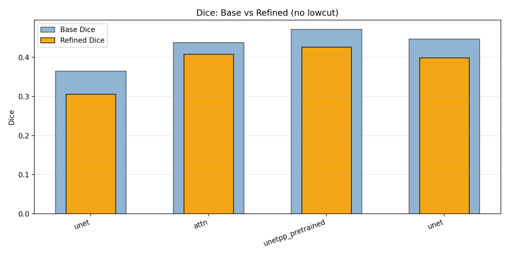


Figure 25: Labeled dataset: Dice, base vs. refined. U-Net's Dice drops sharply with refinement; U-Net++ and Attention U-Net are less affected. U-Net++ Pretrained seems to have the best dice score.

Figure 24 compares MSE between base and refined models. For U-Net, refinement reduces MSE, confirming that the refiner suppresses hallucinations. However, for the other models, MSE either worsens or changes only marginally, suggesting that their cleaner base outputs leave little room for meaningful improvement.

Figure 25 shows Dice scores. Here U-Net++ (pretrained) achieves the highest Dice overall, balancing relatively high recall with fewer hallucinations.

In contrast, refinement generally lowers Dice scores by similar amounts across U-Net++ (base), U-Net++ (pretrained), and Attention U-Net, rather than benefiting any one model. This pattern aligns with the percentage-point TP–FP gap analysis (Figure 19), which showed that refiners consistently remove more true positives than false positives. As a result, Dice tends to decline uniformly across architectures, even if the visual differences between refined and base outputs are subtle.

Together with the precision/recall values in Table 1, these results confirm that U-Net++ (pretrained) provides the most balanced baseline model on the labeled dataset.

Overall, the labeled dataset highlights the trade-off between recall and precision across architectures. U-Net achieves the highest recall but at the cost of very low precision, with the refiner being of no help, since the refiner removes many true positives along with hallucinations. Attention U-Net and U-Net++ are more conservative, yielding higher precision but reduced recall. Among them, U-Net++ (pretrained) stands out: it achieves the best Dice score (Figure 25), maintains relatively low MSE (Figure 24), and balances precision and recall more effectively than the other models. This makes U-Net++ (pretrained) the most reliable baseline on the labeled dataset.

7.3.2 Baseline Model Performance on EHT Data

Table 2 shows the corresponding metrics for the EHT dataset, evaluated at the optimal lowcut per model.

On the EHT dataset, all models perform considerably worse than on the labeled dataset, reflecting both the higher complexity of the images and the poor quality of the annotations, which were produced conservatively by a novice annotator. We return to this issue in Section 7.6.2, where qualitative inspection provides further context for interpreting these results.

Having seen that the labeled dataset already revealed distinct precision–recall trade-offs be-

Table 2: EHT dataset: average TP, FP, FN, and derived precision/recall at each model's optimal lowcut. U-Net recovers far more fibers than others but with low precision, although all models have very low precision. At a glance of this table only, it is clear that U-Net is the best performer by far.

Model	TP	FP	FN	Precision	Recall
U-Net	991	6840	3327	0.13	0.23
U-Net++ (base)	210	1158	4107	0.15	0.05
U-Net++ (pretrained)	117	799	4200	0.13	0.03
Attention U-Net	113	1104	4204	0.09	0.03

tween the architectures, it is useful to now examine how these behaviors manifest on the more challenging EHT data. Here, U-Net recovers the most fibers (recall 0.23) but with very low precision (0.13) due to its extremely high FP rate. U-Net++ (base) slightly improves precision to 0.15 but suffers from very low recall (0.05). U-Net++ (pretrained) and Attention U-Net are the most conservative, with recall near 0.03 but also low precision (0.09–0.13). These results highlight the difficulty of the EHT dataset and the contrasting behaviors of the architectures: U-Net favors recall at the expense of precision, whereas Attention U-Net and U-Net++ sacrifice recall for more conservative, but incomplete, segmentations.

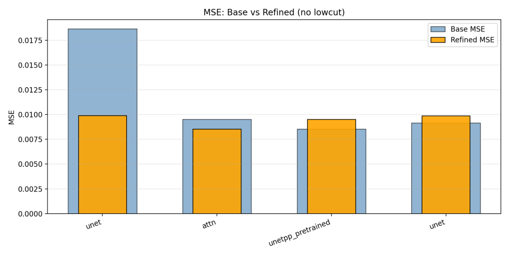


Figure 26: EHT dataset — MSE, base vs. refined. Refinement reduces U-Net's MSE dramatically by suppressing hallucinations; for U-Net++ and Attention U-Net, effects are minor or negative.

Figure 26 shows MSE values for the EHT dataset. Refinement drastically reduces U-Net's MSE, due to suppressing the large number of hallucinations present in the base output. For Attention U-Net and both U-Net++ variants, however, MSE reduces only slightly or even increases, highlighting that refinement fragments already conservative predictions rather than improving them.

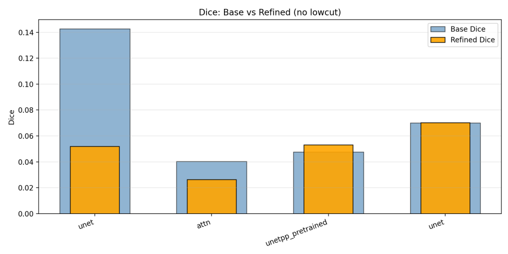


Figure 27: EHT dataset — Dice, base vs. refined. U-Net achieves the highest Dice at its best lowcut (2), but refinement cuts Dice severely. Attention U-Net and U-Net++ achieve modest Dice that remain relatively stable.

Figure 27 illustrates Dice scores. Here, the difference between architectures is striking: the base U-Net achieves by far the highest Dice at its optimal lowcut (2), with values that are more than double those of Attention U-Net and U-Net++ (recall 0.23 vs. 0.03–0.05). Refinement reduces U-Net's Dice sharply, consistent with the recall drop observed in Table 2, but even after refinement it still outperforms the other models numerically. Attention U-Net and U-Net++ variants achieve modest Dice values that are relatively stable with or without refinement, reflecting their conservative behavior: low recall, but also lower FP rates.

In summary, the EHT dataset further illustrates the architectural differences. U-Net achieves significantly higher recall and Dice than any other model, more than four times the recall of Attention U-Net and U-Net++, making it the only model that consistently recovers a substantial fraction of fibers. While its precision is low (0.13), this is not especially worse than the other architectures, which also fall in the 0.09–0.15 range. The key distinction is therefore not that U-Net is uniquely imprecise, but that it achieves much higher recall while the others fail to recover most fibers at all. Attention U-Net and U-Net++ both produce more conservative predictions, reflected in stable Dice values and reduced FP rates, but with negligible recall. Of these, U-Net++ offers slightly better structural fidelity, whereas Attention U-Net is the most resistant to hallucinations. Thus, U-Net provides the most complete but noisy segmentations, while U-Net++ and Attention U-Net prioritize caution but at the expense of missing the majority of fibers.

7.4 Effect of the Refiner Network

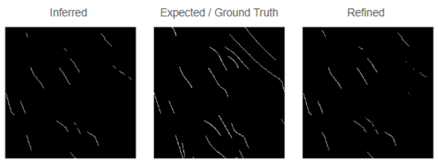


Figure 28: Labeled Dataset: Example of input image, expected outcome, and true output of refiner. This image showscases that the refined output instead of moving closer to the ground truth, primarily removes information from the inferred image.

Figure 28 illustrates the intended role of the refiner network. Comparing the inferred image with the ground truth, it can be seen that some fibers are only partially detected: segments of a fiber appear disconnected, or small false "fibers" are detected that should be removed. The refiner was designed to address these issues by connecting broken segments of true fibers and removing false detections. For example, in the inferred image the rightmost fiber is split into three shorter fragments. The ground truth shows it to be one continuous fiber, so ideally the refiner would join these fragments into a more coherent fiber structure. Importantly, the refiner was not expected to fully reconstruct missing fibers, but rather to enforce local structural consistency.

7.4.1 Refinement on Labeled Data

In practice, the results show that the refiner does not behave as intended. While it sometimes improves structural quality, the quantitative outcomes vary strongly between models. Table 3 summarizes the average changes in TP, FP, and FN after refinement relative to the base models.

For U-Net, refinement reduces false positives by about 260 pixels on average, but this comes at the expense of 172 true positives being lost. The net effect is a lower recall and only a modest precision gain. For Attention U-Net and both U-Net++ variants, the refiner consistently decreases TP (–36 to –56 pixels) while actually increasing FP, making their outputs quantitatively worse. These patterns mirror the Dice and MSE plots for the labeled dataset: refinement generally lowers Dice across all models and only marginally improves MSE for U-Net.

Table 3: Labeled dataset: average change after refinement vs. base outputs. Refiners generally remove more TP than FP, reducing Dice.

Model	ΔTP	ΔFP	ΔFN
U-Net	−172.2	−264.3	+172.2
Attention U-Net	−35.8	+20.5	+35.8
U-Net++ (base)	−56.9	+50.3	+56.9
U-Net++ (pretrained)	−56.2	+61.5	+56.2

This behavior can be explained by the refiner's tendency to remove more than it adds. By aggressively suppressing what it identifies as noise, it lowers FP but simultaneously removes many correctly predicted pixels, increasing FN. The result is that TP decreases across all models, and for some models FP increases. Consequently, Dice scores decline after refinement, but this does not necessarily mean that refined outputs are visually less interpretable. Manual inspection still shows occasional improvements in structural integrity, such as the removal of small hallucinated fragments or smoother fiber shapes, even when overlap metrics worsen.

7.4.2 Refinement on EHT Data

On the real EHT dataset, the refiners behave visually much like on the labeled data, but the quantitative results highlight their much greater potential for suppressing hallucinations in U-Net outputs. Table 4 shows the changes relative to the base models.

Table 4: EHT dataset: average change after refinement vs. base outputs. U-Net gains from strong FP suppression; Attention U-Net modestly improves; U-Net++ increases FP.

Model	ΔTP	ΔFP	ΔFN
U-Net	−790.6	−5260.6	+790.6
Attention U-Net	−52.8	−478.2	+52.8
U-Net++ (base)	+30.4	+517.8	−30.4
U-Net++ (pretrained)	+18.4	+351.8	−18.4

For U-Net, refinement removes a large number of false positives (over 5,200 on average), but at the cost of nearly 800 true positives being lost. This explains the sharp reduction in recall and Dice observed in Section 7.3, even as MSE improves dramatically. For Attention U-Net, the effect is smaller but consistent: around 480 false positives are removed, while just 53 true positives are lost, resulting in only a marginal precision gain.

Interestingly, U-Net++ behaves differently. Both variants actually see a small increase in TP (+18 to +30), but this comes alongside a substantial increase in FP (+350 to +520). Thus, the refiners effectively make U-Net++ more sensitive but less precise, in contrast to their conservative nature. These outcomes suggest that the refiner is especially effective at cleaning up the noisy baseline U-Net, modestly helpful for Attention U-Net, and counterproductive for U-Net++ by reintroducing hallucinations.

Taken together, these results confirm that the refiner is architecture-dependent. On U-Net, it plays an aggressive denoising role, removing thousands of false positives at the expense of recall. On Attention U-Net, it makes only small corrections. On U-Net++, it shifts the balance in the wrong direction, increasing false positives while adding only a handful of true detections. This variability underscores that the refiner cannot be assumed to provide universal benefit: its effectiveness depends critically on the baseline model's error profile.

7.4.3 Thresholding of Refined Outputs

To investigate whether post-processing thresholds could improve the effectiveness of the refiner, we evaluated the outputs at different thresholds $(0.3, 0.5, 0.7, 0.9)$. Table 5 summarizes the results.

Table 5: Labeled dataset: refined outputs across thresholds (0.3–0.9), showing TP and FP averages. Lower thresholds boost recall but also hallucinations; higher thresholds reduce both TP and FP.

Threshold	TP	FP
0.3	424	836
0.5	406	777
0.7	389	722
0.9	362	640

The results show a clear trade-off between true and false positives. At lower thresholds (e.g. 0.3), the models detect more true fibers (higher TP), but this comes at the cost of many additional hallucinations (higher FP). At higher thresholds (e.g. 0.9), false positives are reduced substantially, but true positives also drop, meaning that fewer genuine fibers are recovered.

Intermediate thresholds (0.5–0.7) offer the most balanced outcome: false positives are reduced compared to very low thresholds, while the number of true positives remains reasonably high. For consistency, the 0.5 threshold was used in earlier comparisons between base and refined models. Nonetheless, these results confirm that thresholding alone cannot fully resolve the limitations of the refiner, since suppressing false positives almost always comes with a loss of true positives.

7.4.4 Interpreting Refiner Behavior

The numerical results presented above can be better understood in the context of how the refiner was trained. The refiner learns to map model outputs closer to the true labels, effectively correcting false detections or missing segments. If the base model hallucinates many structures, as in the case of U-Net, the refiner would ideally be trained to suppress them, which explains the dramatic reductions in false positives observed on both the labeled and EHT datasets. Conversely, when the base model is already conservative, as with Attention U-Net or U-Net++, the refiner has less opportunity to remove hallucinations and instead tends to add a small number of detections.

However, across all architectures the refiner demonstrates limited capacity to add true positives. This bias toward removal rather than addition likely arises from the data distribution: positive-class pixels are relatively rare in these images, and during training, suppressing negatives (false positives) may have been the most reliable way to reduce the loss. As a result, the refiner appears to converge toward a strategy of "cleaning up" rather than reconstructing missing fibers.

This behavior shows both the potential and the limitation of the approach. On one hand, the refiner is highly effective at reducing hallucinations, particularly when applied to models such as U-Net that tend to over-predict. On the other hand, it lacks the corrective power to recover missing fibers, which diminishes recall and overall Dice. Future work/improvements may require modified training strategies, such as class reweighting, loss functions that penalize missed positives more strongly, or specialized data augmentation, to hopefully improve the refiners ability to add information.

7.5 Fiber-Level Metrics: Connectivity, Orientation, and Length

To complement pixel-based metrics, we evaluated fiber-level statistics across models: connectivity score, mean angle, and mean fiber length. These metrics provide insight into how well structural properties of the predicted fibers match the ground truth.

7.5.1 Labeled Data

Figures 29–31 show the distributions of length, connectivity, and orientation across models for the labeled dataset.

For **fiber length** (Figure 29), ground-truth annotations display a wide range of lengths with several very long fibers, as expected in the defined labels. U-Net captures some of this range, with longer fibers present in its predictions, though refinement tends to fragment them into shorter segments. Attention U-Net and both U-Net++ variants consistently underestimate length, producing fragmented fibers that rarely approach the lengths of the ground truth. This confirms that the main challenge for these models is not detection of fibers itself but rather the preservation of its continuity.

For **connectivity** (Figure 30), U-Net significantly overshoots, producing highly inflated connectivity scores with wide variance and many outliers. This can be explained by its tendency to hallucinate fibers, which artificially create numerous junctions. Refinement reduces connectivity drastically, closer to ground-truth levels but at the expense of recall. Attention U-Net and U-Net++ predict more conservative structures, with low connectivity values that reflect their underestimation of fiber intersections.

For **orientation** (Figure 31), all models align relatively well with the ground truth: median fiber angles and spreads are similar, indicating that the models are detecting the correct directional trends even when length and connectivity differ substantially. This suggests orientation is the most reliable structural metric across models, whereas length and connectivity reveal their biases (U-Net exaggerates, U-Net++/Attention U-Net underestimate).

Overall, these results show that U-Net preserves continuity but hallucinates, inflating structural metrics, while Attention U-Net and U-Net++ avoid hallucinations but produce fragmented outputs. Orientation is preserved across all architectures, reinforcing that directionality is easier to capture than continuity.

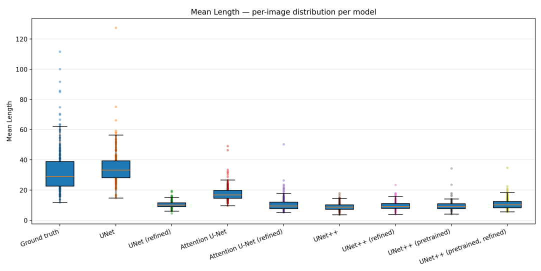


Figure 29: Labeled dataset: mean fiber length. U-Net captures some longer fibers, while other models underestimate length consistently.

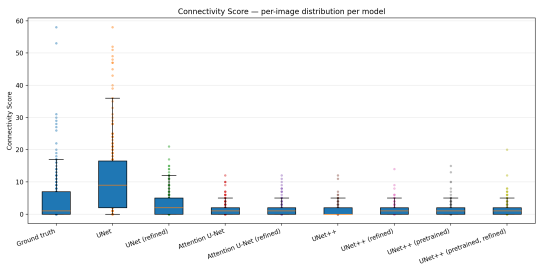


Figure 30: Labeled dataset: connectivity score. U-Net overshoots due to hallucinations, while Attention U-Net and U-Net++ underestimate connectivity.

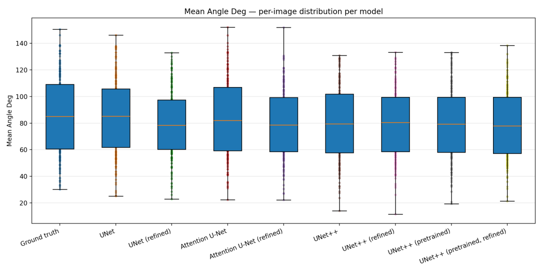


Figure 31: Labeled dataset: mean orientation angle. Orientation is captured well across all models, even where length and connectivity diverge.

7.5.2 EHT Data

Figures 32–34 show the same fiber-level metrics for the EHT dataset.

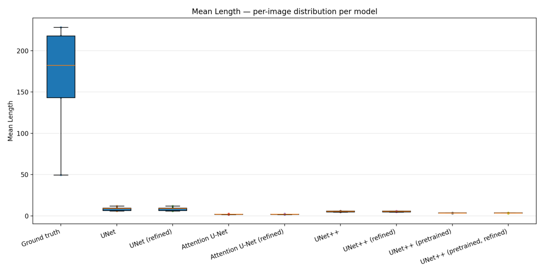


Figure 32: EHT dataset: mean fiber length. All models predict extremely short fibers compared to ground truth, consistent with fragmented outputs.

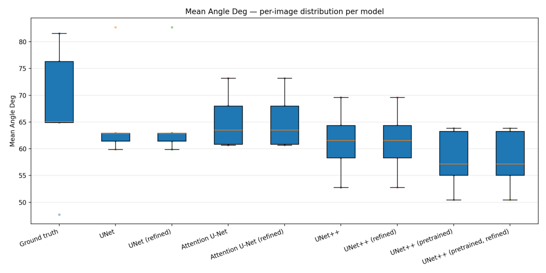


Figure 34: EHT dataset: mean orientation angle. Orientation distributions appear reasonable, but fragmentation undermines global directionality.

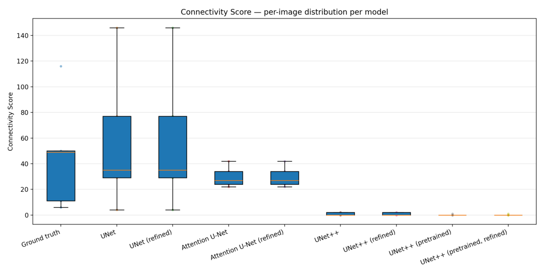


Figure 33: EHT dataset: connectivity score. U-Net's mean aligns with ground truth but with wide variance due to hallucinations; Attention U-Net surprisingly matches well, while U-Net++ consistently underestimates.

For fiber length (Figure 32), the gap between models and ground truth is significant. All models predict extremely short fibers, far below the range of annotated values. This aligns with the disjoint predictions observed earlier: fragmented outputs cannot approximate the long continuous fibers in the ground truth. Given that even on the labeled dataset models underestimated fiber length, this result is expected to be amplified on the more difficult EHT data.

For connectivity (Figure 33), disjoint predictions again lower connectivity scores. Interestingly, Attention U-Net and base U-Net produce mean connectivity values that are closer to the ground truth, though for very different reasons. Attention U-Net aligns surprisingly well, producing a distribution with a similar mean and a narrow spread, whereas base U-Net matches the mean but with an extremely wide spread, likely due to hallucinations generating hallucinated junctions. Thus, the apparent similarity in averages is misleading: Attention U-Net may occasionally approximate true connectivity, while U-Net's mean may simply be a result of its hallucinations. U-Net++ variants consistently underestimate connectivity, producing sparse and fragmented structures.

For orientation (Figure 34), distributions remain relatively close to the ground truth. Median values and spreads suggest that models can capture directional information even when fiber continuity is poor. However, because many fibers are predicted as short fragments, these orientation estimates may not represent the true global fiber organization. Thus, while angle distributions appear numerically reasonable, they dont represent the loss of structural context caused by fragmentation.

Together, these results highlight the interdependence of fiber-level metrics. Short, disjoint fibers lead directly to poor connectivity scores and undermine the reliability of angle measurements. While the labeled dataset provides a clearer demonstration of model capabilities, on the EHT dataset the metrics must be interpreted cautiously, as annotation noise and fragmented outputs obscure the models' true performance. Still, the consistency of angle distributions across datasets suggests that orientation may remain the most transferable metric, even when completeness and connectivity fail.

7.6 Human Inspection of Model Outputs

While quantitative metrics provide valuable insights into pixel accuracy, fiber length, and connectivity, they cannot fully capture how the segmentations appear to a human observer. Certain aspects of structural coherence, perceptual continuity, and visual stability are not represented by numerical scores. To address this, we include a qualitative comparison of model outputs in Figure 35 for the labeled dataset and

Figures 36-40 for the EHT dataset, highlighting differences in human-perceived quality.

7.6.1 Human Inspection on Labeled Data

From visual inspection, U-Net outputs appear the most human-friendly. The predicted fibers are long, connected, and consistent in appearance, resembling the overall fiber structure of the ground truth. However, U-Net also introduces many obvious hallucinations, which are clearly visible as extra fibers not present in the label. As discussed earlier, this explains its inflated connectivity and length metrics: the long continuous predictions come at the cost of over-structuring and adding non-existent fibers.

Attention U-Net produces visually clean and consistent segmentations. Its fibers are rarely broken and contain few visible hallucinations. However, the fibers it predicts are notably shorter than in the ground truth. Careful comparison shows that they do correspond to real parts of fibers, but they lack the full continuity of the original structures. Thus, the segmentations are visually precise but incomplete.

Both U-Net++ variants generate outputs similar to Attention U-Net. The base U-Net++ recovers some longer fibers, though not approaching the lengths captured by U-Net. The pretrained U-Net++ detects more fibers than the base variant but with less clarity: its predictions appear more disjointed and less visually consistent. This suggests that while pretraining enhances sensitivity, it also introduces additional fragmentation.

Across all architectures, refinement consistently produces visually inferior outputs. The refined versions tend to remove information, resulting in thinner and more fragmented fibers. In some cases, refinement does suppress hallucinations (most notably in the base U-Net), but overall it reduces structural coherence and the impression of continuity.

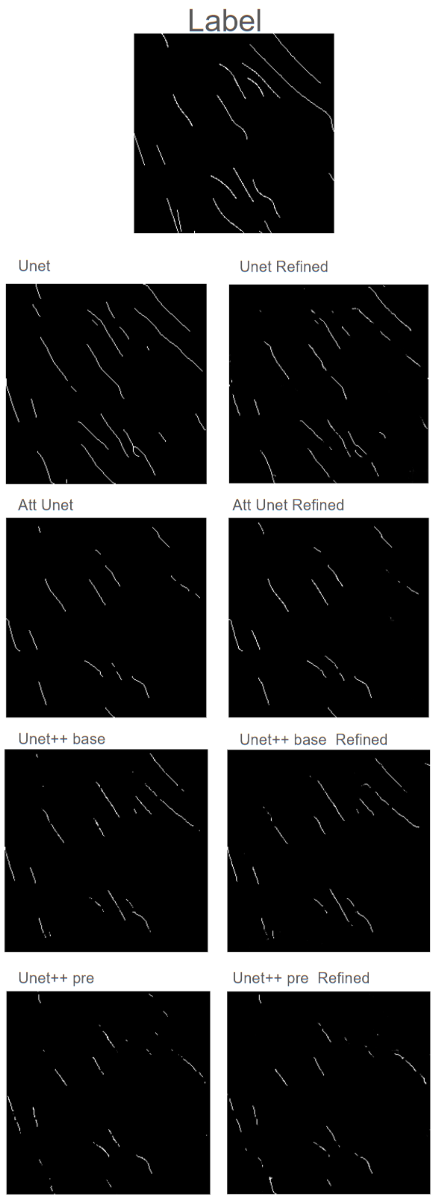


Figure 35: Labeled dataset: qualitative comparison of predictions vs. ground truth. U-Net is most complete but hallucinates; Attention U-Net is clean but incomplete; U-Net++ lies in between; refiners often fragment predictions.

Human inspection therefore confirms and contextualizes the quantitative results. U-Net provides the most visually complete fibers, but exaggerates structure through hallucinations. Attention U-Net yields the cleanest, most reliable outputs, but they are generally shorter than the ground truth fibers. U-Net++ variants lie somewhere in between: the base U-Net++ recovers more fiber structure than Attention U-Net but less than U-Net, while the pretrained U-Net++

produces more fragmented predictions despite higher sensitivity. Refinement generally worsens visual quality, sacrificing completeness and coherence for marginal gains in suppressing false positives.

7.6.2 Human Inspection on EHT Data

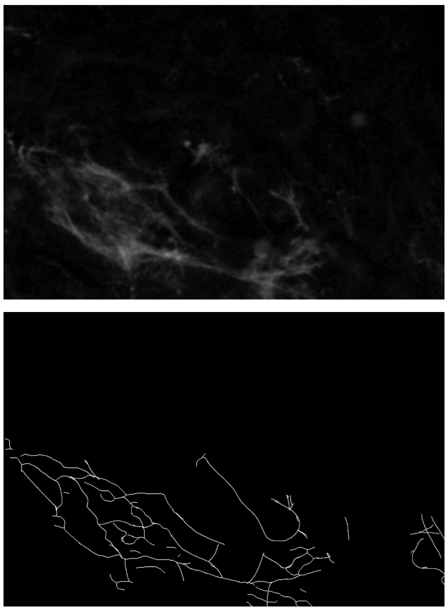


Figure 36: EHT dataset: original confocal slice (top) and its annotation (bottom). Labels omit faint structures, inflating measured false positives for models.

Evaluating the EHT dataset using numerical metrics alone is problematic, primarily due to the quality of the available annotations. As noted earlier, the labels were created by a novice annotator and are highly conservative: only very obvious fibers were annotated, while any ambiguous structures were deliberately excluded. For example, in Figure 36, the above image shows the original EHT slice and the botttom image shows its annotation. Several faint structures visible in the original are omitted from the annotation, even though they may plausibly correspond to fibers. These features become apparent when the image is inspected at its native resolution (4650×2700) on a high-quality display, but cannot be shown clearly here. Models that detect such structures are therefore penalized with false positives (FPs), even though the detected regions may in fact represent true fibers. This suggests that the absolute FP rates reported in Table 2 are likely overestimates, although the relative trends across models remain meaningful.

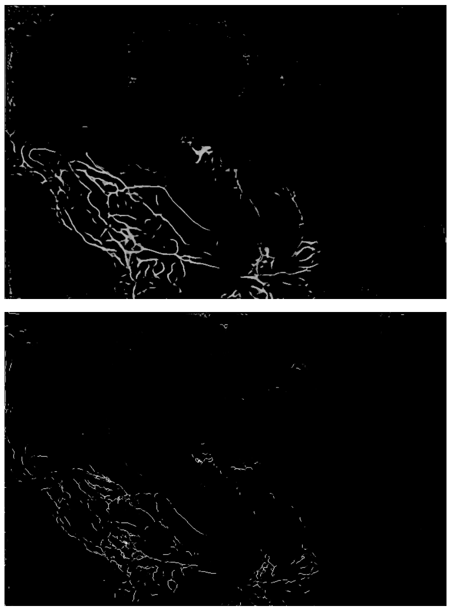


Figure 37: EHT dataset: U-Net predictions, base (top) and refined (bottom). Base captures long fibers but with excessive thickness; refinement thins them but fragments continuity.

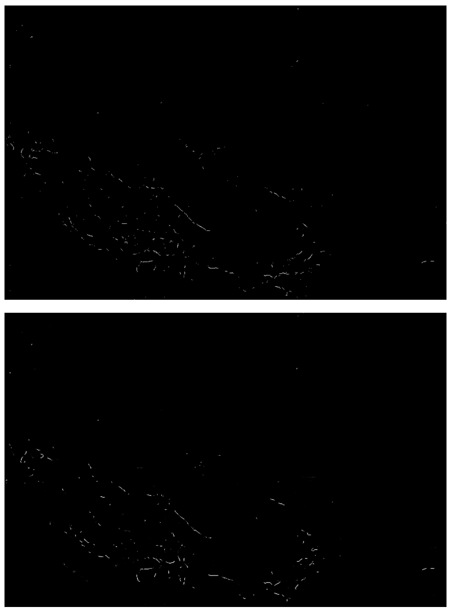


Figure 38: EHT dataset: Attention U-Net predictions, base (top) and refined (bottom). Outputs are clean but fragmented; refinement reduces thickness without restoring structure.

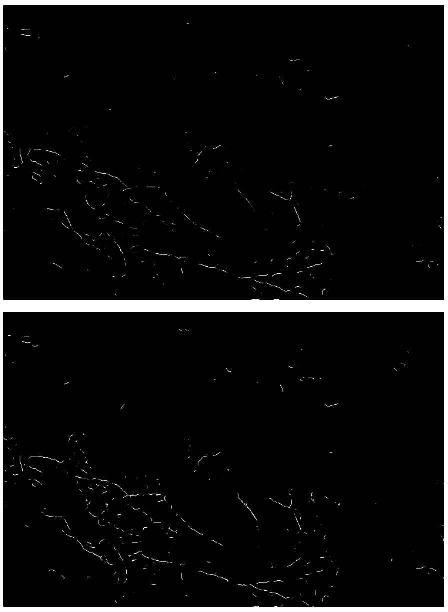


Figure 39: EHT dataset: U-Net++ (base backbone), base (top) and refined (bottom). Base recovers longer fibers than Attention U-Net; refinement often reintroduces false positives.

When visually inspecting the outputs, we observe a more nuanced picture than in the labeled dataset. As shown in Figure 37, the base U-Net recovers the most connected and longest fibers. From visual inspection and comparison to the label and original image, it appears that most of the false positives may stem from the predicted fibers being substantially thicker than the one-pixel annotations, however, we can also see very noticeable hallucinations in the image. This makes the detections look more realistic to the eye, but artificially inflates FP counts. The refiner for U-Net (bottom image of Figure 37) does succeed in thinning these predictions, but at the cost of breaking apart the long, continuous fibers that gave the base model its visual advantage. The result is a fragmented output that resembles the refinements of the other models, however, still preserving a significant amount of the true fiber structure.

Attention U-Net (Figure 38), by contrast, produces short and fragmented predictions, rarely capturing long continuous fibers. Although its quantitative TP rates are similar to U-Net++ (113 vs. 117 on average), qualitatively the difference is clear: Attention U-Net mainly recovers small fiber fragments, whereas U-Net++ (Figures 39–40) captures longer, more coherent stretches of the same fibers. This makes U-Net++ visually more similar to the true fiber architecture despite similar scores.

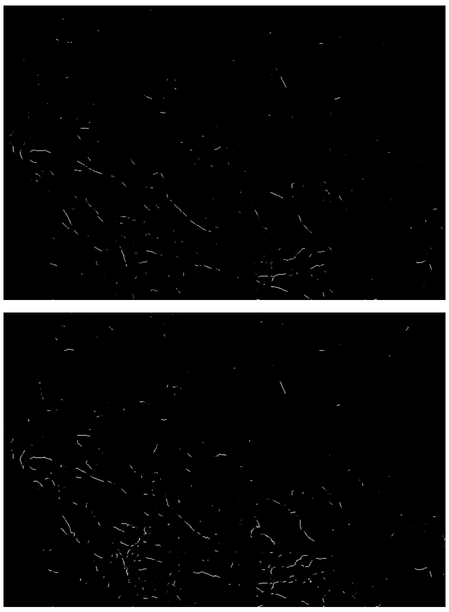


Figure 40: EHT dataset: U-Net++ (pretrained backbone), base (top) and refined (bottom). Pretrained model is more conservative; refinement adds little and sometimes raises false positives.

Comparing the U-Net++ variants, the pretrained version (Figure 40) behaves more conservatively than the base model (Figure 39), often fragmenting fibers further rather than reconstructing them fully. Nevertheless, both U-Net++ variants produce outputs that are more structurally meaningful than those of Attention U-Net, with fibers appearing less like scattered dashes and more like connected fiber elements.

The refiners behave consistently with the labeled dataset: they tend to degrade outputs by disconnecting fibers and removing genuine structures. While they suppress thickness and occasional noise (notably in the base U-Net), this comes at the expense of structural continuity and coherence. As with the labeled data, the refiners show little ability to add missing fiber content.

In summary, visual inspection of the EHT dataset highlights discrepancies with the quantitative metrics. U-Net (Figure 37) provides the most visually convincing reconstructions, with long and connected fibers, but these come at the cost of excessive thickness that drives up FP counts. Its refiner reduces thickness but sacrifices continuity. U-Net++ (Figures 39–40) preserves fiber directionality and length more faithfully than Attention U-Net (Figure 38), which remains the most conservative but also the most incomplete. Overall, if completeness and visual plausibility are valued, U-Net offers the richest representations; if precision and structural fidelity are prioritized, U-Net++ is the stronger choice. The refiners, while useful for suppressing thickness, consistently reduce structural quality, underscoring their limited utility in the current form.

7.7 Summary of Findings

Across both datasets, several consistent patterns emerge. Frequency pre-processing plays a critical role: on the labeled dataset, a moderate lowcut around 5 achieves the best balance between recall and suppression of hallucinations, while on the EHT dataset the optimal setting is architecture-specific (around 2 for U-Net and around 8 for Attention U-Net and U-Net++).

With these settings, U-Net stands out for its high recall, recovering substantially more fibers than any other model, but at the expense of low precision due to thick predictions and hallucinations. Attention U-Net and U-Net++ are more conservative, achieving higher precision but missing many fibers. Among them, U-Net++ (pretrained) provides the most balanced performance on the labeled dataset. The refiner network generally lowers Dice by removing more true positives than false positives, proving useful mainly for suppressing U-Net's hallucinations on EHT, but offering little or even counterproductive benefit for the other models.

Refiner threshold sweeps highlight the expected trade-off: lower thresholds increase recall but also hallucinations, while higher thresholds reduce false positives but miss true fibers.

Fiber-level metrics reflect these trends: fiber length is underestimated due to fragmentation, connectivity is inflated by U-Net's hallucinations and underestimated by conservative models, and orientation is relatively stable but less meaningful when fibers are broken. Human inspection reinforces these conclusions: U-Net produces the most visually complete but noisy segmentations, U-Net++ better preserves structural fidelity with fewer hallucinations, and Attention U-Net is the cleanest but most incomplete. In summary, U-Net is best suited when completeness is prioritized and additional cleanup is feasible, whereas U-Net++ is the stronger choice when precision and structural reliability are valued. The current refiner primarily functions as a cleaner rather than a true reconstructor of missing fibers.

8 Limitations and Future Work

Several limitations of this work should be acknowledged. First, the datasets available were small and not fully representative of real engineered heart tissue (EHT). The labeled dataset lacked the density and structural complexity of true EHTs, while the annotated EHT dataset was limited both in size and in annotation quality, as labels were produced conservatively by a novice annotator. Better annotation practices, ideally involving domain experts and more consistent labeling protocols, would greatly improve the reliability of evaluation and training.

Second, this thesis initially aimed to extract and analyze 3D fiber metrics, since the EHT images are volumetric confocal stacks. However, this proved infeasible in practice. The available EHT volumes contained only $\sim$25 usable z-slices per stack, compared to the very high in-plane resolution ($\sim$9500$\times$5000 pixels), meaning that the data were effectively two-dimensional with limited depth information. Moreover, no suitable 3D training datasets were available to train volumetric models, making robust 3D inference unattainable within the scope of this project. As a result, all analysis was constrained to 2D slice-wise metrics, even though a volumetric approach would be more biologically meaningful. Future work with deeper EHT stacks or realistic synthetic 3D training data could extend this pipeline to volumetric segmentation and metrics.

Third, the refinement networks tested in this thesis often degraded performance, removing true fibers or fragmenting long structures. This suggests that the current design is too biased toward suppressing false positives. Future work should explore alternative strategies, such as reweighted loss functions that emphasize recall, data augmentation to better capture rare positive cases, or entirely different refinement paradigms (e.g. graph-based refiners, transformer- or diffusion-based post-processing, or topology-preserving losses).

Fourth, only a small set of model architectures (U-Net, Attention U-Net, and U-Net++) were explored, with one pretrained backbone (ResNet34). Exploring other pretrained backbones, foundation models for biomedical segmentation, or self-supervised pretraining on large unlabeled microscopy datasets could yield better generalization. Alternative training strategies such as semi-supervised learning or weakly supervised approaches might also allow more effective use of the limited annotations available.

In summary, future work could focus on four main directions: (i) building larger and better-annotated datasets, (ii) extending the pipeline toward volumetric (3D) segmentation and metrics when deeper datasets become available, (iii) investigating more advanced or structurally-aware refinement strategies, and (iv) exploring a broader range of pretrained architectures and training methodologies. Together, these directions could push this pipeline beyond proof-of-concept into a robust and generalizable tool for EHT analysis.

9 Conclusion

This thesis presented a pipeline for automated fiber segmentation and analysis in engineered heart tissue (EHT) images. By combining frequency-based pre-processing, established segmentation architectures, and post-processing refiners, we systematically evaluated the trade-offs between recall, precision, structural fidelity, and visual plausibility.

Across experiments, clear architectural differences emerged. U-Net produced the most complete and connected fibers but at the cost of many hallucinations, while Attention U-Net offered the cleanest predictions but missed much of the fiber structure. U-Net++ provided an intermediate solution, with pretrained weights improving sensitivity but often at the expense of continuity. Refiners, although designed to enforce structural consistency, generally acted as suppressors rather than reconstructors, reducing hallucinations but also removing true fibers. Fiber-level metrics and human inspection confirmed these trade-offs, showing that orientation was captured reliably across models, but length and connectivity suffered from fragmentation or hallucinations depending on the architecture.

Together, these results demonstrate that the pipeline can already extract meaningful structural information from challenging EHT data, despite the limitations of available datasets. U-Net is most suitable when completeness is prioritized, while U-Net++ offers a more precise representation of fiber topology when structural reliability is preferred. Although refiners in their current form provide limited added value, the broader framework shows promise as a foundation for future developments.

References

[1] Client Challenge — pypi.org. https://pypi.org/project/imaris-ims-file-reader/. [Accessed 17-07-2025].

[2] Skeletonize &x2014; skimage 0.25.2 documentation — scikit-image.org. https://scikit-image.org, year = , note = [Accessed 17-07-2025],.

[3] Jeremy S Bredfeldt, Yu Liu, Carl A Pehlke, Mark W Conklin, Patricia J Keely, Thomas R Mackie, and Kevin W Eliceiri. Computational segmentation of collagen fibers from second-harmonic generation images of breast cancer. *Journal of biomedical optics*, 19(1):016007, 2014.

[4] Hao Chen, Xiaojuan Qi, Lequan Yu, Qi Dou, Jing Qin, and Pheng-Ann Heng. Voxresnet: Deep voxelwise residual networks for brain segmentation from 3d mr images. In *Neurocomputing*, volume 312, pages 241–248. Elsevier, 2018.

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

[6] J. Greiner, A.C. Sankarankutty, G. Seemann, T. Seidel, and F. Sachse. Confocal microscopy-based estimation of parameters for computational modeling of electrical conduction in the normal and infarcted heart. *Frontiers in Physiology*, 9, 2018.

[7] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Con-*

ference on Computer Vision and Pattern Recognition (CVPR), pages 5967–5976, 2017.

[8] Hoel Kervadec, Jose Dolz, Jing Yuan, Christian Desrosiers, and Ismail Ben Ayed. Boundary loss for highly unbalanced segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 285–293. Springer, 2019.

[9] Deniz Kobat, Oleg Nadiarnykh, and Nirmala Ramanujam. Deep learning-based segmentation and quantification of collagen fiber networks in second-harmonic generation microscopy images of fibrotic tissues. *PLOS ONE*, 19(1):e0289281, 2024.

[10] Hiba Kobeissi, Javiera Jilberto, M Çağatay Karakan, Xining Gao, Samuel J DePalma, Shoshana L Das, Lani Quach, Jonathan Urquia, Brendon M Baker, Christopher S Chen, et al. Microbundlecompute: Automated segmentation, tracking, and analysis of subdomain deformation in cardiac microbundles. *Plos one*, 19(3):e0298863, 2024.

[11] W. Kowalski, F. Yuan, T. Nakane, H. Masumoto, M. Dwenger, F. Ye, J.P. Tinney, and B. Keller. Quantification of cardiomyocyte alignment from three-dimensional (3d) confocal microscopy of engineered tissue. *Microscopy and Microanalysis*, 23:826–842, 2017.

[12] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *International Conference on 3D Vision (3DV)*, pages 565–571. IEEE, 2016.

[13] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

[14] Hyojoon Park, Bin Li, Yuming Liu, Michael S Nelson, Helen M Wilson, Eftychios Sifakis, and Kevin W Eliceiri. Collagen fiber centerline tracking in fibrotic tissue via deep neural networks with variational autoencoder-based synthetic training data generation. *Medical image analysis*, 90:102961, 2023.

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

[16] Adrien M. Stein, Ayse Can, Ping Lu, Lu Ouyang, Nobuhiro Nishimura, and Peter T. C. So. Deep learning-based collagen fiber segmentation in second-harmonic generation microscopy images of lung tissue. *Scientific Reports*, 11(1):2314, 2021.

[17] Malte Tiburcy, James E Hudson, Paul Balfanz, Susanne Schlick, Tim Meyer, Mei-Ling Chang Liao, Elif Levent, Farah Raad, Sebastian Zeidler, Edgar Wingender, et al. Defined engineered human myocardium with advanced maturation for applications in heart failure modeling and repair. *Circulation*, 135(19):1832–1847, 2017.

[18] Guotai Wang, Wenqi Li, Michael Aertsen, Jan Deprest, Sebastien Ourselin, and Tom Vercauteren. Deep learning for refining medical image segmentation with U-Net variants. *Medical Image Analysis*, 61:101661, 2020.

[19] Sheng Xu, Ying Wang, Jia Zhang, Sean J. Kirkpatrick, and Nirmala Ramanujam. Quantitative analysis of collagen fiber morphology and orientation in shg microscopy images of human breast tissue. *Journal of Biomedical Optics*, 20(8):086005, 2015.

[20] Yueming Zhao, Jing Zhang, Zongwei Wang, and Dimitris Metaxas. Data-driven refinement of coarse segmentation for medical images. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 465–473. Springer, 2019.

[21] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation, 2018.