Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Bendik Austnes

Increasing Validity and Uncovering Utility in Machine Learning Studies

An Illustrative Approach to Essential Concepts and Procedures in Model Development and Assessment

Master's thesis in Cybernetics and Robotics
Supervisor: Adil Rasheed
Co-supervisor: Lise Lyngsnes Randeberg
June 2021

NTNU
Norwegian University of Science and Technology

Bendik Austnes

Increasing Validity and Uncovering Utility in Machine Learning Studies

An Illustrative Approach to Essential Concepts and Procedures in Model Development and Assessment

Master's thesis in Cybernetics and Robotics
Supervisor: Adil Rasheed
Co-supervisor: Lise Lyngsnes Randeberg
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

Recent advances in deep learning has been remarkable. As the availability of computational power and simple-to-use machine learning frameworks are rapidly increasing, deep learning systems are increasingly deployed to new fields of research. Many machine learning studies in medicine report performance comparable or better than clinicians, however many of them were found to be at high risk of bias, and deviated from existing reporting standards. In particular, a frequent lack of evaluation on external data, as well as development on too narrow datasets, limit the medical utility for many of the methods presented. Moreover, classical model development can be time consuming and cumbersome, thus migration to machine learning methods can be tempting. Therefore, there is a need for increased knowledge on the behaviour of machine learning methods among users from non-statistical disciplines, as well as well-defined methods and protocols suitable for machine learning research in various fields. This thesis aims at illustrating the effects and impacts of some important aspects on model development and assessment, in an explanatory and illustrative fashion, striving towards increased understanding and intuition, in order to be more accessible to inexperienced users. Finally, guidelines are presented to assist developers in achieving increased model validity and uncover utility.

Sammendrag

De siste årene har det skjedd store fremskritt innen dyp læring. I takt med stadig økende tilgang på datakraft og brukervennelige maskinlæringsmetoder, har stadig flere forskningsdisipliner tatt i bruk dyp læring. Mange medisinske studier som benytter maskinlæring rapporterer om resultater som er like gode eller bedre enn standard klinisk praksis, men undersøkelser viser at flere av disse studiene kan ha lavere vitenskapelig betydning enn først antatt. Spesielt er det mangel på ekstern validering, samt at flere av publikasjonene baserer seg på svært små og ensformige datasett, som gjør at den faktiske nytten av de nye metodene er usikker. Utvikling av klassiske og veldefinerte metoder kan være svært vanskelig og tidkrevende, noe som gjør at det i mange tilfeller kan være svært fristene å gå over til maskinlæringsmetoder. Derfor er det et behov for å øke forståelse for virkemåten til maskinlæringsmodeller blant brukere innen ikke-statistiske felt, samt å innføre veldefinerte forskningsmetoder og protokoller for å øke validiteten til videre forskning på området. I denne rapporten vises effekten og innvirkningen av sentrale emner innen utvikling og validering av maskinlæringsmodeller. Gjennom en forklarende og illustrativ tilnærming til teori, forsøker rapporten å gi økt forståelse og intuisjon, spesielt for uerfarne brukere. Avslutningsvis foreslås retningslinjer som er ment for å hjelpe utviklere med å oppnå økt ytelse og validitet, og samtidig avdekke vitenskapelig betydning og relevans på nye områder.

Contents

Preface

This thesis is the result of knowledge and experience I have gained throughout my education, and life in general, together with an exploratory approach to problem solving. Starting out with a rather wide problem domain, namely the "improvement in systemic sclerosis diagnostics", I sought for problems that I could solve with my background and experience, and found that I might be able to contribute to the cause with the use of computer vision and data science, topics I have come to enjoy greatly in the past few years. I started working towards model development for semantic segmentation of wrinkles on dorsal finger skin, since the amount and changing characteristics of wrinkles can be linked to diagnostics and progression tracking in systemic sclerosis. The preproject preceding this thesis aimed at uncovering whether a deep learning approach was feasible for wrinkle segmentation at all. The results were promising, and thus the development of deep learning based models using hyperspectral imaging began early this year. Hyperspectral images for wrinkle segmentation is interesting, since it takes on a spatiospectral approach to features of wrinkles, and it is to the length of my knowledge currently unknown whether or not the spectral features can enhance detection in this application.

Unfortunately, due to the COVID-19 pandemic, the hyperspectral imaging lab was closed. The small hyperspectral dataset that was already collected through a previous project featured too low resolution, and thus was not feasible for model development. Infection control restrictions were strongly limiting alternative data collection, but I managed to gather a small high resolution dataset based on ordinary images on a "friends and family" basis for alternative model development. However, validity issues related to very scarce data quickly presented itself, resulting in any exploratory claims essentially being futile. With no possibilities of collecting more data, it was decided to move the scope of the thesis from a *model development* perspective, to a more *explanatory* and *illustrative* perspective, highlighting both obvious issues and pitfalls in machine learning model development, as well as less obvious (but still typical) issues, such as aspects concerning annotation strategies, external validation, etc.

While the change of scope was indeed challenging and frustrating, I learned a great deal from the experience, especially in terms of experimental science in practice, and that sometimes one must quickly adjust to the changing circumstances, find new angles, and make the best out of what one has.

I would like to thank my supervisors Lise Lyngsnes Randeberg, who acted as main scientific supervisor, and Adil Rasheed for guidance and feedback during the project. Acknowledgements also goes to Berit H. J. Grandaunet MD, PhD, for proofreading the medical contents on systemic sclerosis, participants who contributed to the dataset on short notice, and those who was involved in

proofreading and gave comments on the final thesis.

Finally, I would like to thank my friends and family. Special thanks goes to my roommates who keeps me from going insane during this pandemic.

Bendik Austnes
Trondheim, Norway
June 2021

List of Figures

List of Tables

Nomenclature

Medical nomenclature

Crow's feet area	Refers to the wrinkled area that expands from the outward corner of the eyes.
DIP	Distal interphalangeal
Dorsal finger skin	The skin located on the outward facing areas of the finger.
Fibrosis	The process in which connective tissue replaces normal parenchymal tissue to the extent where it is leading to considerable tissue remodelling.
Incidence	The proportion of persons developing a condition during a time period.
Palpation	The process in which the clinician uses her hand to examine the patient's body.
PIP	Proximal interphalangeal
Prevalence	The proportion of persons having a condition during a time period.
RCT	Randomized Controlled Trial
Rhytid	Wrinkle
Sclerodactily	Localized thickening and thightening of the skin.
SSc	Systemic sclerosis
Vasculopaty	A general term to describe any disease affecting blood vessels.

Mathematical symbols

$*$	Convolution operator
$\lfloor c \rfloor$	Floor function. Outputs the the greatest integer less than or equal to a real number c.
$\mathbb{Z}$	Set of Integers
$\mathcal{H}$	Hessian
∇	Gradient
$\odot$	Hadamard product
σ	Standard deviation
s	Size of a Gaussian kernel

Technical nomenclature

(Image) Segmentation	The process of partitioning and image into multiple segments, e.g. wrinkle and non-wrinkle areas.
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DL	Deep Learning
DSA	Digital Subtraction Angiography. Method for visualizing blood vessels.
DSC	Dice Similarity Coefficient, Sørensen-Dice Coefficient
FCNN	Fully-Connected Neural Network. Also called a dense network.
FRF	Frangi filter
GPU	Graphics Processing Unit
HHF	Hybrid Hessian Filter
HLT	Hessian Line Tracking

HSI	Hyperspectral Imaging
i.i.d.	Independently and identically distributed
JSI	Jaccard Similarity Index
LOOCV	Leave-One-Out Cross-Validation
ML	Machine Learning
MR/MRI	Magnetic Resonance/Magnetic Resonance Tomography (Imaging)
PIECES	Protocol Items for External Cohort Evaluation of a deep learning System
RGB	Red-Green-Blue. Usually refers to an image composed of three color channels, one for each color.
SL	Statistical Learning
SWIR	Short wave infrared
TTS	Train-Test-Split. Refers to holdout-set validation.
VNIR	Visual and near-infrared

Chapter 1

Introduction

1.1 Motivation and Background

As many modern science and engineering problems are becoming ever more complex, typically being nonlinear, high-dimensional, and multiscale in space and time [5], classical model development can be cumbersome and time consuming. Sometimes, these systems might not be resolvable by methods based on first-principles [5]. Thus, migration from classical approaches to machine learning approaches can be both tempting and necessary.

With increasing computational power and availability of simple-to-use machine learning (ML) frameworks, the end-users of ML are no longer only centered around the computer science and statistics communities. Tools such as `ImageJ` [6], which is a free software typically used in biology and medicine [7], offers users in these disciplines easy access to various ML classifiers, e.g. for segmentation tasks. Having new disciplines involved with ML can take these methods to fields on which they have never been used before, potentially leading to new breakthroughs. However, as ease of use and access increases, so does the risk of methods being used in questionable ways.

Many ML studies in medicine report performance comparable or better than clinicians [8], however, many of them were found to be at high risk of bias and deviated from existing reporting standards [9]. For medical research, the structure and implementation of randomized controlled trials (RCT) are well established, but no such standardized methods exist for ML and deep learning (DL) studies. In particular, frequent lack of evaluation on external data, as well as development on too narrow datasets, limit the medical utility for many of the methods presented in research [8].

Well-designed methods and protocols are required in the fields of experimen-

tal science in order to achieve validity and uncover utility. As data acquisition and analysis methods varies across fields, specific protocols must be designed, aimed at leveraging benefits of the field to the maximum, while handling associated difficulties. Modern data-driven methods, such as ML/DL, are powerful [10], but lack of interpretability and sensitivity to biased data, as well as mis-handling of data during development and validation, can result in false scientific claims [8; 9].

1.2 Goals and Objectives

Goal *This thesis aims at illustrating the impacts of data leakage, cross-validation, external validation, and the process of manual annotation, in an explanatory and illustrative fashion, striving towards increased understanding and intuition, in order to assist inexperienced users of machine learning and deep learning methods at achieving increased validity and uncover utility of these powerful methods in their fields of study.*

Most users within the field of statistics and data science will be well familiar with the concepts explained and discussed in this thesis. However, from the authors' own experience from several university level machine learning courses, many of the key topics discussed in this thesis are perhaps under-communicated and/or neglected to short side-notes in the lectures. Thus, the target reader for this thesis are inexperienced data analysts and ML users from outside the fields of computer science, statistics, etc.

1.2.1 Objectives

First, the thesis takes the reader through the full development process of a classical computer vision model for wrinkle segmentation. Performance is evaluated and compared to ML models, and it is shown that ease of use and apparent better performance of the ML models, indeed makes migration from a classical approach to a ML/DL approach tempting.

Then, different aspects on model development and assessment of the ML models are investigated in an explanatory and illustrative fashion.

Six objectives are proposed to reach the overall goal of the thesis. Each of the objectives are coupled with guidelines presented in chapter 5, however, in order to fulfill the illustrative qualities, the reader must also consult with the theory, methods and discussion provided in the thesis.

Objective 1 *Illustrate the potentially large impacts from data leakage, and how easily data leakage can occur.*

Objective 2 *Illustrate the importance and benefits of using cross-validation.*

Objective 3 *Discuss external validation, and make the reader aware of the importance having separate development cohorts and external validation cohorts.*

Objective 4 *Explain the benefits of having multiple annotators.*

Objective 5 *Illustrate important aspects of manual annotations in supervised learning, and how they relate to model performance and validity.*

Objective 6 *Explain why failing to clearly state the details of the annotation process can give irreproducible results.*

1.3 Outline of the Report

The report comprises of the following sections and content: Chapter 1 gives motivation and background, goals and objectives. Chapter 2 gives an introduction to the anatomy of the human hand, properties of human skin, previous methods on wrinkle analysis, methods in classical computer vision, an introduction to deep learning, and theory on training and assessment strategies and related topics. Chapter 3 gives the concrete method for devising three algorithms for wrinkle segmentation; one based on classical computer vision, and two based on deep learning based computer vision, where the two models are developed using different training and assessment strategies. Chapter 4 presents the results, gives a discussion on theory, methods and results, and dissects the properties of the datasets, as well as the benefits of cross-validation, impacts on manual annotations, and provides an outlook on the design of deep learning studies in medicine. The report is concluded in chapter 5, and relevant guidelines are presented.

Hyperspectral imaging (HSI) and a HSI dataset are presented in appendix A. Extra material supporting the thesis is given in appendix B.

Nomenclature is given on pp. xvii – xix, where technical abbreviations, mathematical symbols, and a medical dictionary are included.

Chapter 2

Theory

In this section, a basic introduction to relevant human anatomy is given, as well as previous work on wrinkle analysis, general filters and techniques in classical computer vision, introduction to neural networks, and an introduction to training and assessment strategies.

The following sections of this chapter were also presented in the pre-project; 2.2, 2.3.1, 2.3.2, 2.3.3, 2.3.4 (updated), 2.3.8, 2.4.1 (updated and extended), 2.4.2, 2.4.3, 2.4.4 (updated).

2.1 Human Skin and Basic Anatomy of the Hand

This section covers a very brief and basic introduction to human skin and the anatomy of the hand. Also, a very brief introduction to a disease affecting the wrinkles, systemic sclerosis (SSc), is given.

Human skin consists of three layers of tissue, from outermost to innermost; the epidermis, the dermis and the subcutis. The dermis is a fibrous layer, consisting mainly of collagen, that supports and strengthens the epidermis. A network of elastic fibres in the dermis help keep the skin sufficiently tight. The epidermis is the outermost layer of the skin. Epidermal cells are mostly forming in the bottom part of the epidermis where they are in contact with the dermis, before they gradually ascend to the surface and eventually die [11].

The visual appearance and color of the skin are partly due to blood in superficial vessels (e.g. if a person blushes), but mainly due to melanin, a pigment manufactured among the basal cells of the epidermis [11]. The *in vivo* absorption spectrum of melanin is present in the whole visible range (400-720 nm), and is exponentially increasing in the blue–violet wavelength range (400-500 nm) [12].

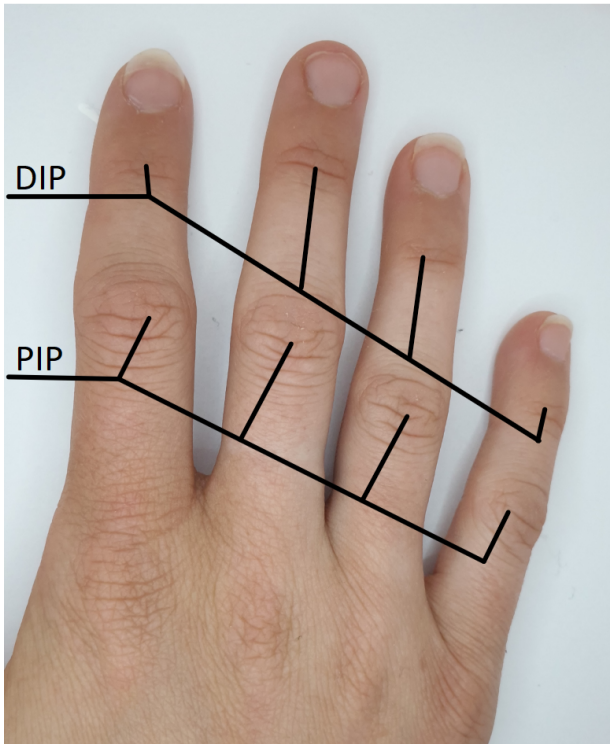

Figure 2.1.1: Image of the right hand of a healthy female. The proximal interphalangeal (PIP) and distal interphalangeal (DIP) joints are indicated in the figure.

The human hand consists of several bones and joints, and with significant variations in the structural and functional characteristics of the skin. The skin of the palms are thicker and more rugged than that of the backs of the hands and fingers [11]. The skin surrounding the proximal interphalangeal (PIP) and distal interphalangeal (DIP) joints, indicated in fig. 2.1.1, need to be especially flexible in order to not prevent smooth joint movement.

Systemic Sclerosis (SSc) is an autoimmune rheumatic disease with relatively low incidence and prevalence. The disease is characterized by excessive production and accumulation of collagen, vasculopathy and immunological abnormalties [13]. The accumulation of collagen, called fibrosis, can prohibit smooth movement of joints due to decreased flexibility of skin surrounding the joint. The origin of the disease is unknown [14]. The hand of a SSc patient is shown in fig. 2.1.2. Note the low amount of wrinkles over the DIP joint in the figure due to sclerodactily.

Figure 2.1.2: Image of the right hand of a SSc patient. Notice that the skin is very smooth from the PIP joint and outwards. Wrinkles are diminishing in the same areas. Source: Maria Sieglinda von Nudeldorf, published under licence CC BY-SA 4.0, via Wikimedia Commons.

2.2 Wrinkle Analysis: Previous Work

Wrinkle analysis typically refers to wrinkle segmentation. A typical application is age estimation. Few methods exist. Frangi proposed a filter for enhancing vessels [15]. The Frangi filter (FRF) uses second order derivatives for ridge detection. A drawback of the FRF for forehead wrinkle detection is that it is omni-directional; it segments both horizontal and vertical discontinuities as wrinkles [1].

Cula et al. [16] developed a method for wrinkle detection by estimating the local dominant direction of elongated spatial features in a neighborhood around each pixel in the image. Then, a Gabor filter with a fitting frequency and the angle set to match the local dominant direction around the current pixel was used for post-filtering. The method was especially focused on the crow's feet area. In the original paper, images were captured using high quality photography equipment in a controlled environment for the purpose of using those images for wrinkle detection algorithms.

Hybrid Hessian Filtering (HHF) was first proposed by Ng et al. to enhance the FRF for wrinkle analysis [1; 17]. Elbashir et al. [18] coined HHF to be considered state-of-the-art in facial wrinkle assessment. [1] proposed a new method, Hessian Line Tracking (HLT), for improving the HHF. The HLT is a seed-based method; its end result is dependent on the number of start seeds of the algorithm. If the number of start seeds are too low, the algorithm will under-segmentate wrinkles, since not all wrinkles are reachable from the given seeds. In the opposite condition, where the number of start seeds are too high,

the algorithm will over-segmentate since start seeds may encounter significant non-wrinkle discontinuity lines in the image.

The aforementioned methods all focus on 2D image data. A 2019 publication by Decenciére et al. [19] is based on wrinkle segmentation by adopting a 3D point cloud captured by fringe projection into 2D topographic maps, and applying morphological openings and closing to the image. The openings and closing are determined by assumed width, depth and directions of the wrinkles to be detected. The results are promising for major wrinkles.

2.3 General Filters and Techniques

In this section we present general filters and techniques used in the project. These methods are not made specifically for wrinkle analysis and segmentation.

2.3.1 Frangi Filter (FRF)

The Frangi filter (FRF) was originally intended for segmentation of vessels in 2D/3D images of various medical imaging modalities such as digital subtraction angiography (DSA), rotating X-ray and MRI in coronary angiography. The algorithm searches for tubular-like geometric structures. For deriving the local principal direction at a point x_0 at scale s (the size of a Gaussian kernel), the second order Taylor expansion in the neighborhood of x_0, given by [20]

$$L(x_0 + \delta x_0, s) \approx L(x_0, s) + \delta x_0^T \nabla_{o,s} + \delta x_0^T \mathcal{H}_{x_0,a} \delta x_0 \tag{2.3.1}$$

is used, where differentiation is defined as a convolution with derivatives of Gaussians:

$$\frac{\delta}{\delta x} L(x, s) = s^\gamma L(x, s) * \frac{\delta}{\delta x} G(x, s) \tag{2.3.2}$$

where γ is a normalization constant and the $\mathcal{D}$-dimensional Gaussian is defined as

$$G(x, s) = \frac{1}{\left(\sqrt{2\pi s^2}\right)^{\mathcal{D}}} e^{-\frac{||x||^2}{2s}} \tag{2.3.3}$$

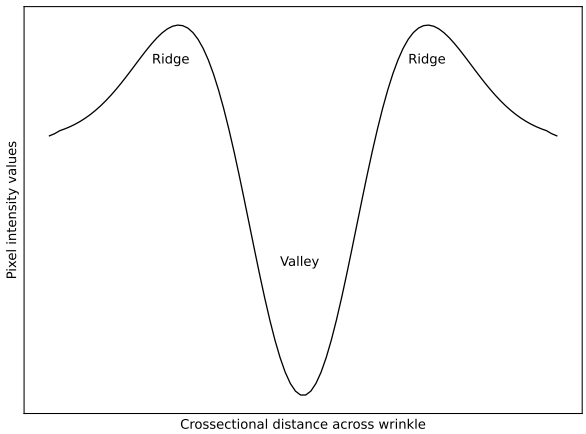

Figure 2.3.1: The second order derivative of the 1-dimensional Gaussian ensembles the ideal wrinkle profile of the pixel intensity values of a crossection perpendicular to the local wrinkle direction. Ridges and valley are indicated in the figure. Figure adapted from [1].

Thus, the Hessian is a matrix of second-order derivatives of Gaussian kernels. These kernels are called probe kernels, and measure the contrast between regions inside and outside the range $(-s, s)$ [20]. When sweeping these probing kernels over an image, the kernels will output a maximum value when they are centered on over an area of values of similar shape as the probe kernel. The eigenvalues of the Hessian give the direction of maximum curvature. Since the Hessian is derived with various Gaussian kernel sizes, we record for which scale the probing kernel gave the maximum output. Then the FRF uses this information in order to give the direction along the smallest curvature, that is, along the vessel (or wrinkle [1]), for the scale s of best fit.

Consider the FRF for wrinkle analysis: if we consider the cross section of a wrinkle perpendicular to the local wrinkle direction, we see that the pixel intensity values ensemble the second order derivative of a 1-dimensional Gaussian kernel with ridges and valley. This is shown in fig. 2.3.1.

2.3.2 Gabor Filter

The Gabor filter is a linear filter used for texture analysis. The filter is a Gaussian kernel modulated by a sinusoidal plane wave [21]. It is used for extracting textures with specific frequencies in specific directions. The filter is specified by a frequency ω and a direction θ. When a Gabor filter is convoluted with an image, the output will highlight geometric structures in the image with the approximate frequency ω and approximate direction θ. A Gabor kernel with $\omega = 0.1$ and $\theta = \frac{\pi}{2}$ is shown in fig. 2.3.2. We can also interpret the frequency

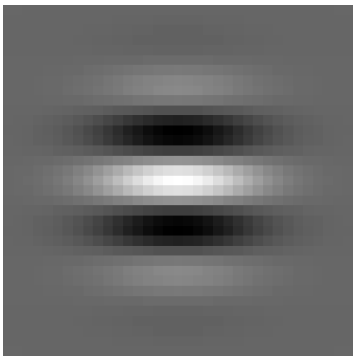

Figure 2.3.2: Gabor kernel with $\omega = 0.1$ and $\theta = \frac{\pi}{2}$.

parameter ω as a size metric. For a Gabor kernel with constant size, lower frequency amounts to the extraction of larger geometric structures.

2.3.3 Hysteresis Thresholding

Hysteresis thresholding is a two-stage method. The algorithm takes two input parameters: the strict threshold limit `high` and the slack threshold limit `low`. In stage 1, the algorithm searches for all pixels that take values above `high`, and adds them to the current segmented set. In stage 2, using all the pixels from stage 1 as seeds, it searches for all other pixels that take values above `low`, and that is connected to another pixel in the current segmented set.

This technique is very useful as a post-processing filter, if the previous filter segmented values from the background by adding a ridge between the segments and the background, i.e. the segments and the background take the same pixel values.

Consider fig. 2.3.3. The hysteresis threshold algorithm will start at the brightest points on the horizontal lines, and build the segmented set outwards into values that take the same value, or even take smaller values than the background. This is possible as long as the desired lines are separated by dark ridges, which prohibits the hysteresis algorithm of moving into the background. It is important to note that for sufficiently low slack threshold values, the algorithm can move out of the ridge boundaries if the ridges do not form a closed set around the desired segments with either another ridge or the image boundaries.

Figure 2.3.3: Gabor filtered image. The desired lines we wish to segment is separated from the background by dark ridges.

2.3.4 Contrast Histogram Equalization

Contrast histogram equalization is the process of changing the pixel intensity values in the image such that after the transform, each pixel intensity has the same number of occurrences. The method is useful when the whole image takes values of approximately the same brightness. By histogram equalization, the global contrast of the image is usually increased [22].

The algorithm is often used as a step in the preprocessing, when the successive algorithms depend on uniform contrast levels in the dataset [22].

2.3.5 Anti-Aliasing: Gaussian Blur

Gaussian blurring is a method for anti-aliasing images. The input image is anti-aliased by convolving the image with a Gaussian kernel. Convolving a Gaussian kernel on the image amounts to low-pass filtering the image, thus reducing the image high-frequency components. The cut-off frequency is selected by adjusting the size of the Gaussian kernel.

2.3.6 Morphological Transformations

Morphological transformations involves techniques for analysis and processing of geometrical structures. It is based on set theory, lattice theory, topology and random functions. There are two basic operators in morphology; dilation and erosion. Then, combinations of dilation and erosion form morphological openings and morphological closings. This section is based on [23].

Morphological transformations in its simplest nature takes a binary input image

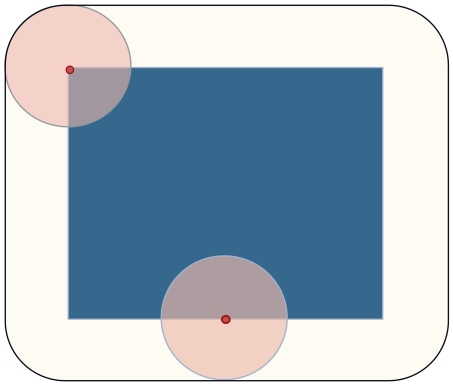

Figure 2.3.4: Dilation. A circular kernel is applied to an input image (blue rectangle) with morphological dilation, resulting in the larger light box with rounded corners. Figure adapted from Renato Keshet (Wikimedia Commons) [2].

and a kernel. The input image is the image on which we wish to apply our transform. The kernel decides the specifics on how the operator transforms the image.

Dilation

Dilation takes in an input image and a kernel. The kernel may be of any shape, but is shown in fig. 2.3.4 as a disk dilating a blue rectangle, resulting in a larger light box with rounded corners. The basic operation of dilation is that the kernel slides through the input image, limited by the kernel's center of gravity as indicated in the figure with red dots. Then any pixel which lies under the kernel at any point gets the value of 1, while all other points remain unchanged.

The method is effective at filling holes in the foreground, for example pepper noise or other discontinuities. It is also effective at re-joining broken elements; e.g. if a line has a hole in it, creating two disjoint lines, the dilation can rejoin the lines (if the kernel is large enough).

Erosion

Erosion is sometimes considered the opposite of dilation. A kernel slides through the image. Then, a pixel is given the value of 1 only if all pixels under the kernel in the original image have a value of 1, otherwise it is set to 0 (it is eroded). Erosion is illustrated in fig. 2.3.5 where a blue rectangle is eroded by a circular kernel, resulting in a smaller light rectangle.

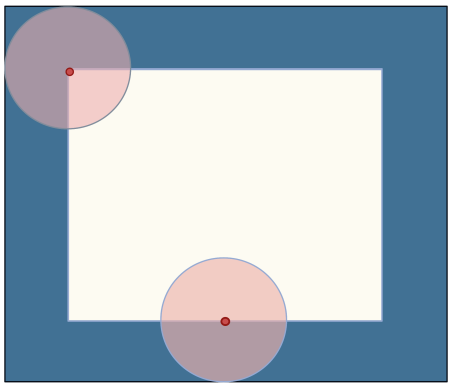

Figure 2.3.5: Erosion. A circular kernel is applied to an input image (dark rectangle) with morphological erosion, resulting in a smaller light rectangle. Figure adapted from Renato Keshet (Wikimedia Commons) [2].

Closing

A morphological closing is a dilation followed by erosion.

Dilation fills holes in the foreground and rejoins discontinuities. However, the dilation results in the foreground dilating. Thus, erosion is applied with the same kernel as that of the dilation, effectively trimming back the foreground roughly to its original size, while still retaining the enhancements provided by the dilation (filling holes and rejoining discontinuities).

Opening

A morphological opening is an erosion followed by dilation. It can be used for noise removal, usually in the background.

2.3.7 Elastic-, Grid- and Optical Distortion

These image distortion transforms can be helpful during data augmentation when dealing with non-rigid structures that have shape variations, which is often the case in medical imaging [24].

Elastic Transforms

Elastic distortion was used by Simard et al. [25] to vastly expand the MNIST [26] dataset. The idea was to apply transformations corresponding to random oscillations of the hand muscles when writing numbers, damped by inertia. The method involves computing a new target position for a point (x, y) wrt. the previous position. The new target position is denoted $\Delta x(x, y)$ and $\Delta y(x, y)$ for x and y, respectively.

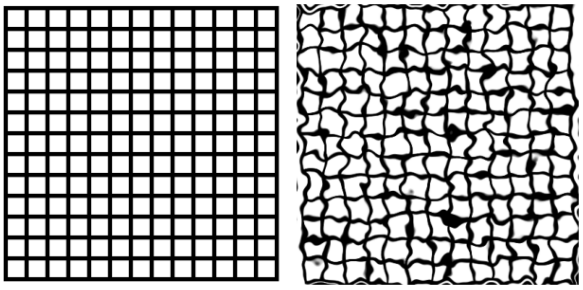

Figure 2.3.6: A square lattice grid (left) transformed by elastic distortion (right).

The elastic distortion is created by first generaing random displacement fields $\Delta x(x, y) = \text{rand}(-1, +1)$ and $\Delta y(x, y) = \text{rand}(-1, +1)$, where $\text{rand}(-1, +1)$ is a random number in $[-1, 1]$ drawn with a uniform distribution. The fields are then convoluted with a Gaussian kernel of size s, where s is the standard deviation (in pixels). Finally, the displacement field is normalized.

s (or σ which is frequently used in literature [25]) is called the *elasticity coefficient*. A small s gives completely random directions, while a large s gives a displacement field close to affine[1]. However, intermediate values of s results in what is called elastic distortions [25], in which lines and parallelism are slightly preserved, making the output image number recognizable, with the desired effects of random oscillations of the hand muscles.

Grid Distortion

Grid distortion involves stretching and squeezing the input image along the horizontal and vertical axes while maintaining image dimensions [24; 27]. Unfortunately, [24; 27] does not provide us with a well defined description of the transform, and we were not able to find a clear description elsewhere. However, fig. 2.3.7 gives us a good intuition on the behaviour of the transform.

Optical Distortion Transform

The optical distortion transform is used to emulate optical distortions. Optical distortion is the situation in which physically straight lines are bended and deformed due to optical aberrations, making them appear curvy [28]. Figure 2.3.8

[1] Affine transformations are transformations that preserves lines and parallelism.

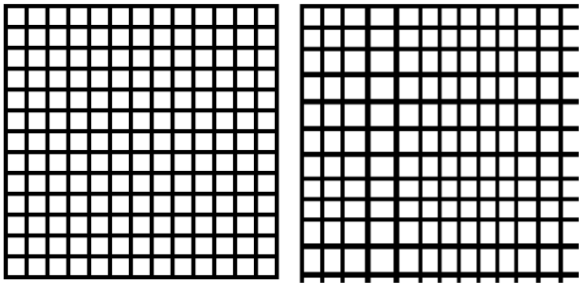

Figure 2.3.7: A square lattice grid (left) transformed by grid distortion (right).

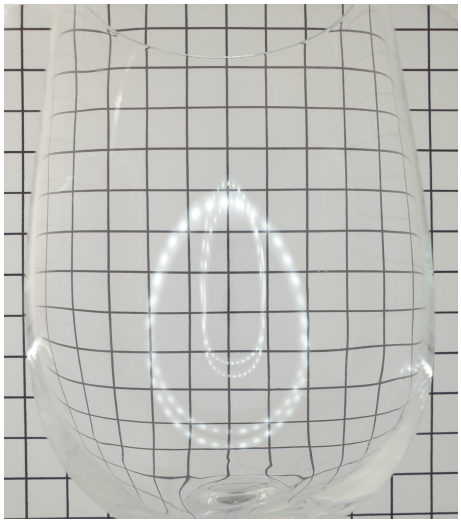

Figure 2.3.8: A square lattice grid subject to optical distortion when being photographed through a wine glass. The ellipsoid-like reflections are specular reflections from the light source.

shows how the square lattice grid being distorted when photographed through a wine glass.

2.3.8 Sørensen-Dice Coefficient. Jaccard Similarity Index

The Sørensen-Dice coefficient ("Dice similarity coefficient (DSC)", "Dice") is a similarity measure independently developed by Sørensen (1948) and Dice (1945). The coefficient is defined as [29; 30]

$$DSC = \frac{2|A \cap B|}{|A| + |B|} \tag{2.3.4}$$

where A and B are sets and $|A|$ is the cardinality of the set.

The Sørensen-Dice coefficient is very similar to the Jaccard Similarity Index (JSI), which is given by [31; 1]

$$JSI = \frac{|A \cap B|}{|A \cup B|} \tag{2.3.5}$$

2.4 Artificial Neural Networks and Deep Learning

Modern computer vision is largely based on the application of artificial neural networks (ANNs) and deep learning (DL). A special type of ANNs is called convolutional neural networks (CNNs). CNNs offer weight sharing, which reduces the number of trainable parameters and thus decreases training time. Furthermore, CNNs offer the property of spatial invariance. This is a key property in CNN based computer vision, since it allows the network to classify objects independently of its position in the input image.

In this section we give a short introduction to fully-connected neural networks (FCNNs), CNNs and autoencoders. Finally, we introduce the U-Net, which is a CNN autoencoder developed for image segmentation on medical images. The content in this section is roughly based on Nielsen [32], Goodfellow [33] and Ronnerberger [3].

2.4.1 Fully-Connected Neural Networks (FCNN)

The Fully-Connected Neural Network, also called a "dense" network, is the most basic network type used in deep learning. It is built up by several nodes, or perceptrons, inter-connected in a net-like structure.

The Perceptron

The perceptron, also called a *node* or a *neuron*, is the fundamental building block of the FCNN. It is illustrated in fig. 2.4.1 with three inputs and three outputs in layer L, however, the perceptron itself is independent of the number of inputs and outputs.

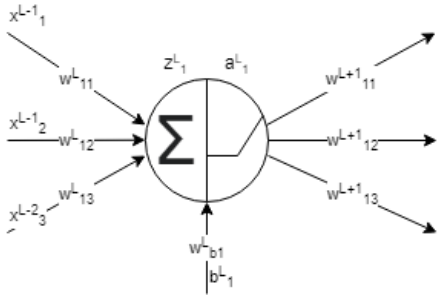

Figure 2.4.1: The perceptron illustrated with three inputs x_{1i}^{L-1}, input weights w_{1i}^{L}, summation of inputs z_1^L and activation a_1^L. Furthermore, the input weights w_{1i}^{L+1} belonging to a potential next neuron is shown by w_{1i}^{L+1}. $i \in \{1, 2, 3\}$. The bias with its corresponding weight is used to adjust the activation threshold.

The perceptron performs two essential tasks; the summation of input values with input weights, and passing the result through an activation function.

The summation of input values with input weights are given by

$$z_1^L = \sum_{i=0}^{n^{L-1}} \left(x_i^{L-1} w_{1i}^L\right) + b_1^L w_{b1}^L \tag{2.4.1}$$

where $i \in [1, n^{L-1}]$ denotes the number of input nodes from layer $L-1$ to layer L. Each input value is multiplied with a corresponding weight w_{1i}^L. Finally, a bias is multiplied with its corresponding weight to control the activation threshold.

Then z_1^L is passed through a non-linear activation function. The activation function determines the *activation* of the perceptron. Several activation functions exist, for example the sigmoid $a^L(z^L) = 1/(1 + e^{-Z^L})$, the tangens hyperbolicus $a^L(z^L) = \tanh z^L$ and the rectified linear unit (ReLU) $a^L(z^L) = \max(0, z^L)$. The sigmoid and the tanh activation functions are commonly called "squashing functions", since they take any values in $\mathbb{R}$ and squeeze them into a smaller range. The ReLU on the other hand just makes sure the perceptron is not activated for negative z^L values, thus introducing an activation threshold that needs to be overcome in order to activate the neuron.

Multilayer Perceptron (MLP)

When two or more perceptrons are inter-connected in series, we have a multilayer perceptron. This is what is commonly referred to as a FCNN or a dense

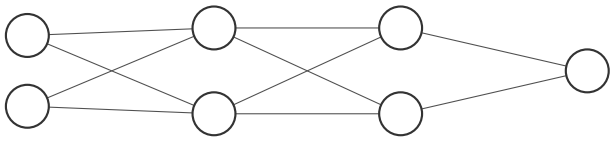

Figure 2.4.2: A 3-layer multilayer perceptron. The input layer has no weights and no activation, therefore it is not counted as a layer. For each perceptron in the hidden layers and the output layer, the ordering of inputs, weights, summation and activation are as shown in fig. 2.4.1.

neural network. An example of a 3-layer MLP is shown in fig. 2.4.2. The input layer has no weight and no activation function, therefore it is not counted as a layer. For each perceptron in the hidden layers and the output layer, the ordering of inputs, weight, summation and activation are as shown in fig. 2.4.1.

If we input values at the input layer, the values will be multiplied with the weights between the input layer and the next hidden layer. Then, in the perceptrons in the hidden layer, the inputs are summed and passed through an activation function, and output to the next hidden layer. The same happens in the output layer, although the output layer typically has an activation function that is very specific to the output we want. For example, if we are trying to predict a value that can take any number, we cannot use a sigmoid activation function in the output layer since this would limit our predictions to values in range $[0, 1]$.

When training the network, the basic idea is that we start off with a network with randomly initialized weights. The first sample will thus output something completely random. Then the output is compared to a label, the "ground truth", through a cost function. The cost function outputs the loss based on how far from the label we predicted. Then we compute a gradient for updating our weights in order to minimize the loss on the next run.

Cost Function and Loss

The cost function is a function in which we input our labels y and predictions $\hat{y}$, and determine the loss. The loss is thus a measure of "how far" our predictions are from the truth. Several cost functions exist, and they can be tailored to fit very specific problems [34; 35].

A typical cost function is given by the Mean Squared Error (MSE). It is given as [32]

$$C(w) = \frac{1}{2N} \sum_{n=1}^{N} (y^n - \hat{y})^2 \tag{2.4.2}$$

where N is the number of training samples, and we divide by 2 so that we can remove that constant when we differentiate later.

If we look at one single training sample, we can write the cost function as

$$C^n(w) = \frac{1}{2}(y^n - \hat{y})^2 \tag{2.4.3}$$

Forward Pass

The forward pass is the process of inputting values to our network, and processing the values as they propagate through the network. For each layer l we compute the input of the node by

$$z^l = W^l a^{l-1} + b^l \tag{2.4.4}$$

where a^{l-1} is the activations from the previous layer $l-1$, W^l is the weight matrix for inputs at layer l and b^l is a column vector of biases for the nodes in layer l.

Backpropagation

The backpropagation, commonly shortened to "backprop", is the process in which we after a forward pass go backwards in the network to find the gradient for minimizing the loss from our cost function. We now show the backprop algorithm on the network shown in fig. 2.4.2 with ReLU activations in the hidden layers and linear activation in the output layer. The backprop algorithm consists of four steps outlined by Nielsen [32]:

Compute error δ^L for layer L (the last layer)

$$\delta^L = \nabla_a C \odot f'(z^L) \tag{2.4.5}$$

where $\odot$ denotes the Hadamard product, and the gradient for one training sample is given by (omitting superscript n from eq. (2.4.3))

$$\nabla_a C = \frac{\partial C}{\partial a_j^L} = -(y - \hat{y}) = -(y - a^L) \tag{2.4.6}$$

where the latter equality states $\hat{y} = a^L$ since we have linear output activation. Expanding eq. (2.4.5) for our network we get

$$\delta^3 = -(y - a^3) \odot f'(z^3) \tag{2.4.7}$$

Use δ^l to compute error in next layer in backwards order The equation is given as follows:

$$\delta^l = [(w^{l+1})^T \delta^{l+1}] \odot f'(z^l) \tag{2.4.8}$$

Bias. The gradient wrt. the bias of node j in layer l is given by [32]

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \implies \frac{\partial C}{\partial b^l} = \delta^l \tag{2.4.9}$$

where we prefer to use the latter expression since it is denser and we do not need to worry about specific nodes j, as this will be resolved directly by matrix algebra.

Each weight The gradient for updating each weight is given by

$$\frac{\partial C}{\partial w^l} = \delta^l (a^{l-1})^T \tag{2.4.10}$$

where the right hand side denotes the outer product of δ^l and a^{l-1}.

Weight and Bias Update The final step in the learning algorithm is to update the weight and bias values. The update rule is given by

$$\theta_{k+1} = \theta_k - \eta \frac{\partial C}{\partial \theta_k} \tag{2.4.11}$$

where η is the learning rate controlling how far we move in the gradient direction, and θ_{k+1} is the weights and biases for the next cycle.

Capacity, overfitting and underfitting

The capacity of a model roughly refers to the model's ability to fit a wide variety of functions [33]. While a model with low capacity may struggle to fit a function to the training set (underfitting), a model with high capacity is more prone to overfitting, since it may memorize a mapping from input to output in the training set instead of finding general structures. Very high capacity models can perform perfectly in the training set. Strongly overfitted models are likely to perform worse on data outside the training set due to the noise utilized to fit the training set [8].

Increasing model capacity generally amounts to increasing the number of internal model parameters, i.e. increasing the number of hidden layers or perceptrons in each layer.

Regularization

Regularization refers to all measures taken to avoid overfitting [33]. Several approaches exist. Two examples of widely used methods, are the dropout algorithm and $\mathcal{L}_p$-regularization.

The **dropout algorithm** is a simple, yet powerful algorithm. For each training step, randomly pick $n = \lfloor pN \rfloor$ perceptrons where N is the total number of perceptrons in the network and p is the proportion of perceptrons we want to pick. Then for all the n perceptrons chosen, set their corresponding weights to zero. This effectively removes these perceptrons from the network. We can look at a large network as a multiple of several smaller networks. Hence, as all these smaller networks are learning features of the input data, the combination of all these predictions may lead to overfitting. When weights are set to zero, the remaining perceptrons can be viewed as a "thinned" version of the original network [36].

It has been shown that dropout improves the performance of neural networks on supervised learning tasks in several disciplines, including computer vision and speech recognition [36].

Another example of a regularization technique is $\mathcal{L}_p$**-regularization**, p integer. $\mathcal{L}_\mathbf{p}$-regularization is done by penalizing the cost function by the $\mathcal{L}_p$-norm of the weights. Typically, the $\mathcal{L}_1$-norm (taxicab norm) or $\mathcal{L}_2$-norm (Euclidean norm) are used. While $\mathcal{L}_1$-regularization typically yields sparse weight matrices, $\mathcal{L}_2$-regularization yields smaller weight values.

2.4.2 Convolutional Neural Networks (CNN)

A convolutional neural network (CNN) is a special type of a neural network. It is commonly used in computer vision. At the very bottom, the CNN is comparable to the MLP; it has weights, activation functions, cost functions etc. However, a major difference occurs in the topology of the network as well as the way it uses weight sharing. It is spatially invariant to the location of objects.

Consider the 4×4 greyscale image represented in matrix form

$$X = \begin{bmatrix} 1 & 3 & 1 & 6 \\ 2 & 4 & 1 & 3 \\ 6 & 5 & 5 & 2 \\ 1 & 3 & 1 & 4 \end{bmatrix} \tag{2.4.12}$$

We want to pass this image through a CNN. For simplicity, we use the convolutional kernel

$$W^L = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.4.13}$$

which is denoted W since the convolutional kernel contains the weights of the network. CNNs have several parameters, among them we have the `stride` and `padding`. Stride is the number of pixels we move our kernel during one step in the convolution. The padding adds zeros to the borders of the image. In the forward pass, we convolve the input with the convolutional kernel and have `stride=1` and `padding=0`, and for simplicity, we apply the ReLU activation function (which has no effect on this data)

$$a^L = X * W^{L-1} \tag{2.4.14}$$

$$= \begin{bmatrix} 1 & 3 & 1 & 6 \\ 2 & 4 & 1 & 3 \\ 6 & 5 & 5 & 2 \\ 1 & 3 & 1 & 4 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 5 & 5 & 7 \\ 14 & 6 & 8 \\ 6 & 8 & 3 \end{bmatrix} \tag{2.4.15}$$

Note that the image dimensions after the convolution is reduced from 4×4 to 3×3.

Further downscaling of the data can be performed by applying pooling functions, such as the max pooling algorithm (`MaxPool2D`). The `MaxPool2D` is specified with a certain size, for example 2×2. If such a pooling scheme is applied, the pooling kernel will slide over the image, and for each step it extracts the largest value from the input data within the kernel at current position. This is illustrated below with unit stride and zero padding:

$$z^L = \text{MaxPool}\left(\begin{bmatrix} 5 & 5 & 7 \\ 14 & 6 & 8 \\ 6 & 8 & 3 \end{bmatrix}\right) = \begin{bmatrix} 14 & 8 \\ 14 & 8 \end{bmatrix} \tag{2.4.16}$$

Hence, the CNN is effective at reducing the input data dimensions, which is necessary, since images typically consists 1000s of data points. Moreover, the max pooling is a way of providing spatial invariance to the CNN, since it extract the largest value within an area regardless of where in that area that value is located. Other types of pooling also exists, such as average pooling.

As previously mentioned, the kernels contain the weights in the CNN, thus as training proceeds in a similar manner to that of the MLP, the kernels change.

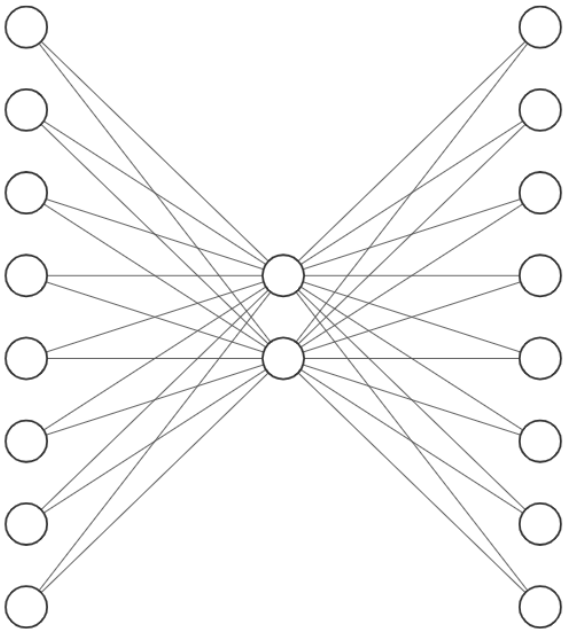

Figure 2.4.3: An autoencoder with 8-dimensional input and 2-dimensional bottleneck. The leftmost part of the network is called the encoder, and the rightmost part of the network is called a decoder.

For CNNs applied on image data, some of the kernels typically obtain similar structures of commonly known filters used in classical computer vision, such as Gaussian blur, Gabor filters etc. Hence we can regain the properties from classical computer vision into our CNN. Other kernels may take on other values, more specific to the data we are training on, which can enhance classification and prediciton properties compared to a classical approach.

2.4.3 Autoencoders

The autoencoder is a type of neural network that is used to learn a sparse representation of the input data and reconstruct the reduced dimension data back to our original data. An illustration of a basic autoencoder that inputs 8-dimensional data, passes it through a 2-dimensional bottleneck and then tries to reconstruct the 8-dimensional data from the 2-dimensional sparse representation is shown in fig. 2.4.3.

A CNN autoencoder has a similar structure, except that the encoder and decoder are switched out with a CNN instead of a FCNN. The leftmost part of the autoencoder is called the encoder, and the rightmost part is called the decoder. The autoencoder typically have identical but flipped architecture in the encoder and decoder parts.

2.4.4 U-Net: Convolutional Networks for Biomedical Image Segmentation

The U-Net [3] is a CNN autoencoder developed for segmentation of the membrane in the Drosophilia first instar larva ventral nerve cord (VNC). It won the ISBI cell tracking challenge 2015.

The architecture of the U-Net is illustrated in fig. 2.4.4 and consists of a contracting path moving downwards (encoder) and an expanding path moving upwards (decoder). The encoder consists of repeated applications of two 3×3 convolutions, followed by the ReLU activation function and a 2×2 max pooling operation with `stride=2` for downsampling [3]. At each downsampling step, the number of feature channels (filters/kernels) are doubled. Similarly, in the decoder, the U-Net maintains a similar structure, however, the convolutions are replaced with up-convolutions and max pooling is changed with up sampling operations. At the final layer, a 1×1 convolution is used to map the 64-component feature channels to the number of classes (membrane vs. non-membrane, etc) [3]. At each convolution in the decoder, the feature map is copied and concatenated with the feature map after the corresponding upsampling stage in the decoder. The network uses unpadded convolutions, hence the feature map dimensions of after the convolution and the corresponding upsampling no longer matches. The U-Net simply crops the larger dimension to make it fit [3]. The original paper claim that this technique is effective for precise spatial localization of points in the image.

The original paper used the cross entropy loss function, since it provides the ability for multi-label classification. The batch size was set at 1 sample. Their input data consisted of 30 samples of 512×512 32-bit grayscale clinically annotated images. Data augmentation, the process of transforming copies of images in the dataset (making random crops, rotations, elastic distortions etc) was used to extend the dataset. The network was trained on a Nvidia Titan GPU (6 GB) for 10 hours [3].

2.5 Training and Assessment Strategies

In this section, we give the overall goals of proper model assessment, as well as how different training and assessment strategies influence the expected performance and feasibility in obtaining good estimates of the true generalization error. First, an introductory section gives a basic understanding of important terminology. Then, two popular training and assessment strategies are presented. Finally, a short introduction to the *Curse of Dimensionality* is given to enhance our perspective on dataset sizes.

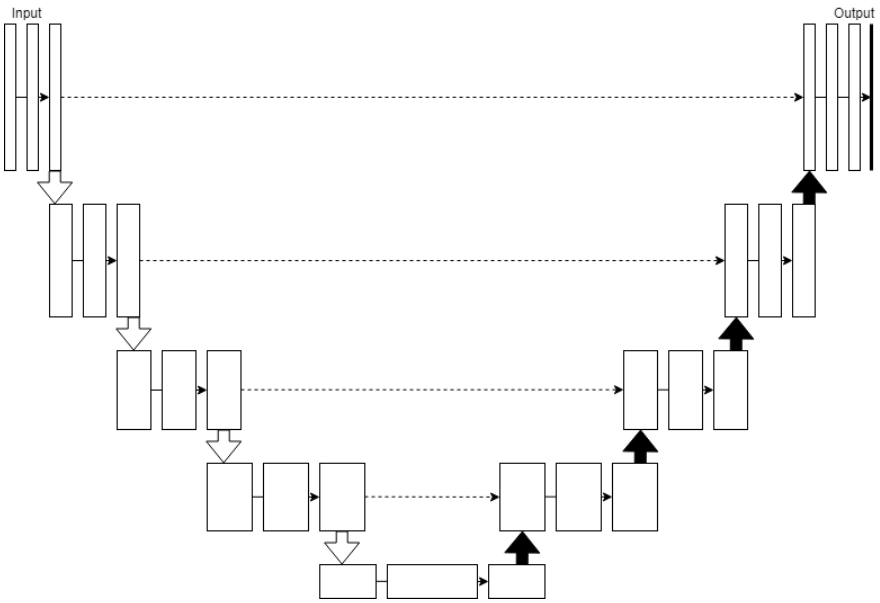

Figure 2.4.4: Illustration of the U-Net. Solid horizontal arrows are convolutions or up-convolutions, downward pointing thick arrows are max pooling, upward pointing thick solid arrows are up sampling, horizontal stippled arrows are the copy, crop and concatenation paths for each level in the encoder/decoder. Illustration adapted from the original paper [3]. The network takes an input image, and outputs a segmentation map for each pixel in the image containing the predicted class.

2.5.1 Assessment of Machine Learning Models in a Nutshell

A machine learning (ML) model contains internal model parameters which needs to be estimated to give a final model. In case of a neural network (NN), weights and biases are adjusted by the backpropagation algorithm as shown in section 2.4.1. This process is called training. For each training cycle, the model is subject to a new set of data, and the parameters are adjusted to improve model performance.

The goal of a ML model is to generalize from the samples it was exposed to during training, to real-world data. In case of the cat/dog classification problem, featured in Kaggle's Dogs vs. Cats dataset [37], we want the model to generalize the overall fundamental structures of cats and dogs. If the model generalizes well, the model will be good at distinguishing between the two animals, even if it is exposed to a an input image that it has never seen before.

Several metrics exists for measuring the performance of models. Accuracy is typically used in classification problems, where a score of 1 is given in case of a correct classification, and 0 otherwise. Hence, the average accuracy over a set of samples is given by the number of correct classifications divided by the

number of samples it was tested on [33]. In segmentation problems, other metrics, such as the Jaccard similarity index (JSI) of the Sørensen-Dice coefficient (DSC) is commonly used [3; 16]. The error, or error rate, is some form of error measurement measuring how good a prediction is. The error metric could be as simple as $1 - \texttt{accuracy}$ for classification problems, or a more complicated error metric involving a special loss function $l(\mathbf{y}, \hat{\mathbf{y}})$, specially chosen for the current problem. In this section, the exact error metric is not important, and we let E denote some error metric that is relevant for a given problem.

When a model is created, we want to know how well it generalizes to the data in a population. The generalization error E_{gen} denotes the true error of the model if it is exposed to all samples in a population. Finding the true generalization error is often infeasible – in case of the cat/dog classification problem, this translates into testing the model on every single cat and dog that exists. This is obviously an impossible task to achieve. However, we aim at estimating E_{gen} by other methods. The typical estimation technique involves testing the model on a sufficiently large well-balanced and unbiased test dataset [33]. The data samples in the test set must be independently and identically distributed (i.i.d.) randomly drawn from the true distribution. The error on the test set is denoted E_{test}, and we expect that $E_{gen} \approx E_{test}$.

The test set should be sufficiently large, well-balanced and unbiased. In case of the cat/dog classification problem, this essentially means that the test set must include a sufficiently large number of different breeds, images must be taken in various situations, and of course, the proportions of cats and dogs must be equal. The number of samples required is strongly dependent on the variation in the population. For a cat/dog classification problem, a rather large test set is required since there exists several thousand different breeds of both cats and dogs. On the other hand, if we are working on data with a more narrow distribution, a smaller number of samples in the test set may suffice.

At this point, we need to make a few notes on model behaviour on previously unseen data, before we continue our introduction on estimation of generalization error. Some model parameters, typically known as hyperparameters, are parameters that we do not want the learning algorithm to optimize. An example of such parameters, are parameters that control model capacity. If the learning algorithm is allowed to optimize on capacity parameters, e.g. number of hidden layers in a NN, it will always choose values that give maximum model capacity [33], resulting in overfitting.

The first time a model is evaluated on the test set, the test set gives an unbiased estimate of the generalization error. However, if we test the model, and then adjust the model to improve performance, E_{test} is no longer unbiased, since it has been indirectly used in selecting model parameters. The situation can be viewed as an "outer" training loop – the inner training loop being the training

cycle where the internal model parameters are being estimated, and the outer training loop being the developer adjusting hyperparameters and network architecture based on test results. Hence, the test set has indirectly become part of the training set. This situation is called *data snooping* or *data leakage*. And since adjustments are being made to improve performance of the model, not only is E_{test} biased, but it grows ever more optimistic[2] for each outer training loop cycle we perform.

Still, we need to be able to adjust model behaviour during development. Since the model is fitted to the training data, we cannot make design decisions based on E_{train}, since this metric underestimates the generalization error (E_{train} is optimistic). The solution is to create a tuning[3] set of data samples that the learning algorithm does not observe [33].

The tuning set is an extract of the dataset on which we do not train the model directly, but we use it to evaluate model performance on unseen data. Thinking back to the outer training loop, this is where the tuning set comes into play. It is a set which is disjoint from the training set, so the model has not seen the data before, and thus it cannot create a mapping from input to output on this data. However, since we are making adjustments to the model based on the tuning results, the tuning error E_{tune} will be optimistic, but usually not as optimistic as E_{train} [33].

Thus, we expect the errors from evaluating with the different sets to have approximately the following relationship:

$$E_{train} \leq E_{tune} \leq E_{test} \approx E_{gen} \tag{2.5.1}$$

Similarly, in terms of a generic performance metric P, where larger is better, we would expect

$$P_{train} \geq P_{tune} \geq P_{test} \approx P_{gen} \tag{2.5.2}$$

However, this is not a rule, and one may very well encounter situations where for example $E_{tune} \geq E_{test}$, and this would not necessarily be a problem. The problem first arises when we have done things that invalidate our estimates of E_{gen}. In the next few sections we give methods for best practice when assessing ML models.

[2] A metric is **optimistic** if the estimation is biased towards better performance.

[3] In this thesis, we favor the term *tuning* over *validation* to avoid naming ambiguity when cross-validation is introduced.

2.5.2 Holdout-Set Validation: Train-Test Split (TTS)

A training and assessment strategy frequently promoted by introductory courses on ML in university, as well as typical ML internet resources, such as `medium.com` and `towardsdatascience.com`, is the train-test-split (TTS). The method has many names; holdout-set validation, independent validation test set, and so on [38; 39; 40]. The main characteristic of the method is that the dataset is divided into two disjoint sets; a training set and a test set. Typically suggested train/test split fractions are 70/30% [5] respectively, or somewhere in that region. The training set is then used to estimate internal model parameters, and the test set is used to test model performance on unseen data, in order to provide an estimate of the true generalization error E_{gen}.

The test set is assumed to be **i.i.d.** drawn from the true population on which we want to make predictions. Hence, the test set has to be sufficiently large in order to capture the variation of the population, and subsequently be useful for estimating E_{gen}.

The amount of samples necessary to obtain a sufficiently large test set are dependent on the variation in the true population. For example, in case of the cat/-dog classification problem, a rather large test set is required to capture the true variation of the population, due to the existence of 1000s of breeds of both cats and dogs. Furthermore, the test set should ideally be well-balanced wrt. classes (e.g. the proportion of cats and dogs should be equal). Re-sampling techniques exist for artificial balancing, but they will not be presented here.

2.5.3 Development Cohort and External Validation Cohort

Development Cohort. Internal Validation.

The development cohort refers to the dataset on which a model is developed. It may be primary data collected by the research team, or secondary data made available to the research team, or public data. During model development, the training, tuning and test set are all subsets of the development cohort. Thus, internal validation refers to model evaluation on the test set contained within the development cohort [8].

External Validation Cohort

External cohorts differ non-randomly from the development cohort [8]. The subjects in the external cohorts are often within a similar population as that of the development cohort, since the model is after all developed for a particular case. However, they must be *external* to the development cohort in some matter. For example, this can be achieved by collecting the external cohort from a

different hospital, or letting the development cohort have one stage of cancer while the external cohort has another stage of the same cancer, using different acquisition equipment (e.g. different camera and light setups), etc. [8].

2.5.4 *k*-Fold Cross-Validation

For small datasets, it may be very hard, and sometimes impossible, to obtain a test set with an identical distribution to that of the true population; in order to increase the similarity of the distribution, a bigger portion of the dataset can be held out for testing. But this can subsequently lead to the training set loosing its similarity to the true distribution. Thus, for small datasets, withholding data from training can be wasteful and leading to a less effective prediction model than it otherwise could be [41]. Furthermore, a small test set implies statistical uncertainty in the estimated generalization error [33]. Increasing the test set, results in decreasing the training set; hence it increases the expected generalization error. Thus, we are standing in the crossroads between achieving minimum expected generalization error (having a large train set), and being able to estimate the generalization error (having a large test set) [41]. We cannot have both, since the sets are disjoint.

In order to improve efficacy of a model developed on a small dataset, while still retaining the possibility of estimating generalization error, various types of cross-validation can be used. In this section, we will be concerned with k-fold cross-validation.

A typical k-fold cross-validation works as follows: consider fig. 2.5.1. We first divide the dataset into k subsets of equal size. With $k = 3$, the method is called 3-fold cross-validation. Then, one subset is selected as validation fold, and the two remaining subsets are merged together to form a training fold. This selection procedure continues until all $k = 3$ subsets have acted as validation sets.

The model is trained and validated k times, each time with a new training fold and validation fold. The cross-validation error $E_{val,cv}$ can be reported as the average of all k validation errors [33],

$$E_{val,cv} = \frac{1}{k} \sum_{i=1}^{k} E_{val}^{i} \tag{2.5.3}$$

where E_{val}^{i} is the validation error at fold i.

If k-fold cross-validation is used without a separate test set, then $E_{val,cv}$ can be used to estimate generalization error E_{gen} [33], *if and only if* no model

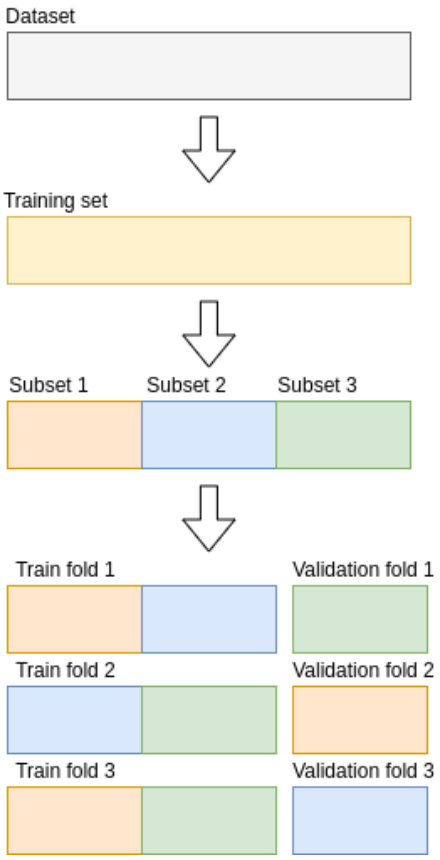

Figure 2.5.1: 3-fold cross-validation.

adjustments was made based on validation results. If adjustments have been made, $E_{val,cv}$ is biased (and usually optimistic).

Choosing k

Choosing the number of training folds k is largely dependent on the size of the dataset. Goodfellow [33] states that expected generalization error can never increase as the number of training samples increases, and that the expected generalization error decreases asymptotically until it reaches the *Bayes error*, or until the model saturates due to capacity restrictions, whichever comes first. This situation is illustrated idealized in fig. 2.5.2. Please note that the numeric values of dataset size in the figure are for explanatory purposes only, and do not constitute a general relationship between generalization error and those specific dataset sizes. In terms of k-fold cross-validation, each training fold can be viewed as to be its own dataset, and so the relationship between generalization error and training fold size follows the same idealized relationship as given in the figure.

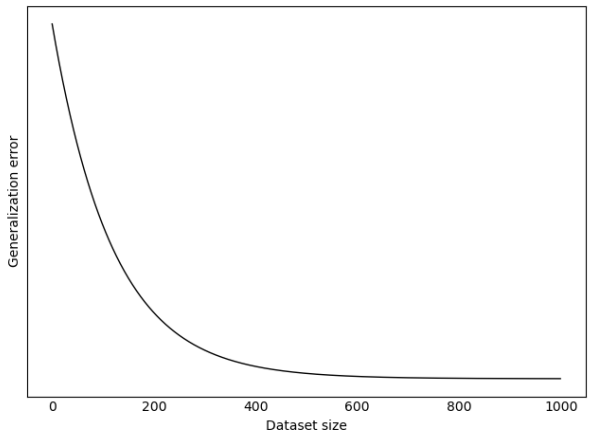

Figure 2.5.2: Illustration of the idealized relationship between generalization error and dataset size. A larger dataset typically yields better generalization properties from the model. Note that the numeric values of dataset size are for explainability purposes only, and does not constitute a typical relationship between datasets of those actual sizes and generalization error, since this relationship is strongly dependent on the distribution of the true population from where the data originates from. Figure adapted from Bjarne Grimstad [4].

Example 1. Suppose your full training set contains $n = 1000$ samples. If you choose $k = 5$ folds, each subset contain $\frac{n}{k} = 200$ samples, yielding a training fold of $\frac{n(k-1)}{k} = 800$ samples. From fig. 2.5.2 we see that the generalization error when training on 1000 samples vs. training on 800 samples are approximately the same. Thus, for this dataset, $k = 5$ may very well be an appropriate value.

Example 2. Now, suppose your full training set contains only $n = 200$ samples. If you choose $k = 5$ folds, each subset contains $\frac{n}{k} = 40$ samples, yielding a training fold of $\frac{n(k-1)}{k} = 160$ samples. From fig. 2.5.2 we see that the generalization error when training on 200 samples vs. training on 160 samples varies quite a bit. Thus, for this dataset, $k = 5$ may retain too much data from the algorithm. In order to increase the amount of available training data, we can increase k. With $k = 10$, the training folds contain 180 samples, which gives lower generalization error.

Leave-One-Out Cross-Validation (LOOCV)

In the most extreme version of k-fold cross-validation, we let $k = n$, where n is the size of the dataset. This is called Leave-One-Out Cross-Validation (LOOCV). This method minimizes expected generalization error wrt. fig. 2.5.2.

The price of increasing k in order to minimize expected generalization error is increased computational costs. The model needs to be trained k times. Thus, in the case where LOOCV is used, we need to train our model once for each sample in the dataset. For a large dataset, this strategy may be completely infeasible – and unnecessary. However, for small datasets, this may very well be a necessary price to pay in order to minimize generalization error while maximizing efficacy.

2.5.5 The Curse of Dimensionality

The curse of dimensionality is the name of the phenomena where machine learning problems become exceedingly difficult as the number of dimensions in the data increases [33]. We will present this phenomena with a simple example. Consider the cat/dog classification problem. First, assume that we have converted our dataset into grayscale images of cats and dogs. With this setup, the ML algorithm does not need to take color into consideration when learning features, thus it will focus on spatial structures from the get-go. However, if we change to a colored dataset, the algorithm also takes this into consideration. Since cats and dogs can take on "any" color, the algorithm now needs to learn whether or not color has anything to do with the final classification or not. At the same time, it has to learn spatial features. Thus, adding the color feature to our cat/dog classification problem significantly increases the amount of training samples required from the algorithm to learn the fundamental structures between the two animals.

Adding features also impact the balance in the dataset. With a colored cat/dog dataset, we need to make sure that the dataset is balanced wrt. color, otherwise the ML algorithm may overestimate certain categories based on occurrence frequency, e.g. if there are a lot of black cats, and not many black dogs, the algorithm might learn that black is a *fundamental structure* of a cat, which we know it is not.

Chapter 3

Methods and Setup

This section introduces the methods, models, and setup used in our development and assessment of models for semantic wrinkle segmentation. First, two datasets are presented. Then, a method based on classical computer vision methods which we name the Frangi-Gabor process is given. Next, we present the U-Net CNN autoencoder, immediately followed by two U-Net models featuring different training and assessment strategies.

Relevant hardware, software, drivers, as well as ethical consideration and special COVID-19 considerations are given in section 3.5.

3.1 Datasets

3.1.1 The Kumar Dataset

The dataset used in the initial development of the Frangi-Gabor process and semi-supervised learning with U-Net is the Finger Knuckle Skin dataset by Ajay Kumar [42]. It will hereafter be referred to as the "Kumar dataset".

The Kumar dataset consists of middle finger dorsal skin images of 503 individuals, where each individual were photographed either 4 or 5 times with short time intervals in between (seconds to minutes), yielding a total of 2515 samples. Some individuals were also photographed after a longer time period (4–7 years). These images was not used in this project. For each sample, the proximal interphalangeal (PIP) and distal interphalangeal (DIP) joint finger dorsal skin (see fig. 2.1.1) have been sorted into separate folders of image dimensions 160×180 px. One PIP image from each individual was extracted. The participants in this dataset are mainly staff and students at Hong Kong Polytechnic University and IIT Delhi Campus, thus 88% of the participants are below the age of 30. The gender distribution is unknown. Furthermore, the majority of the

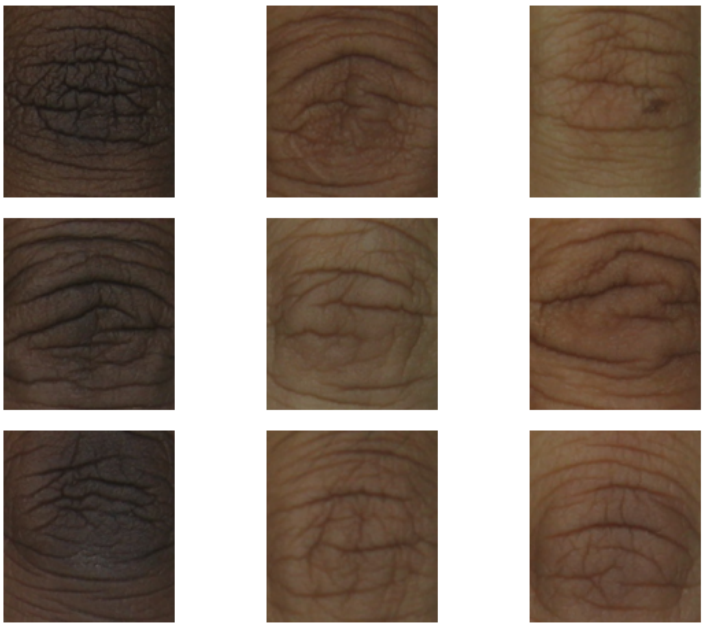

Figure 3.1.1: A preview of the Kumar dataset.

participants are medium dark or dark skinned. A preview of the Kumar dataset is shown in fig. 3.1.1.

3.1.2 High Resolution Dataset (HiRes Dataset)

High resolution images of 12 individuals was collected. The individuals in this dataset are assumed to be healthy – a quick examination by palpation was performed as well as a self declaration from the participants confirming they did not have any known medical conditions. The number of participants were limited by the ongoing covid-19 pandemic.

The average birth year of the participants was $\mu_{birthyear} = 1996.7$ with standard deviation $\sigma_{birthyear} = 2.9$. The youngest participant was born in 2000 and the eldest participant was born in 1991. The male/female ratio was 0.67.

A custom made box for taking the images in homogeneous light conditions, as well as maintaining fixed distance and angle from the camera to the participant's hand, was used during data collection. The box is shown in fig. 3.1.3. The box was spray painted white on the inside to increase uniform lighting conditions. Furthermore, a hole in the box from where the camera would see through a "selfie ring" (see table 3.5.2) for minimization of shadows in the wrinkles was made.

The camera used was the back-side camera of a Samsung Galaxy S9 smart-

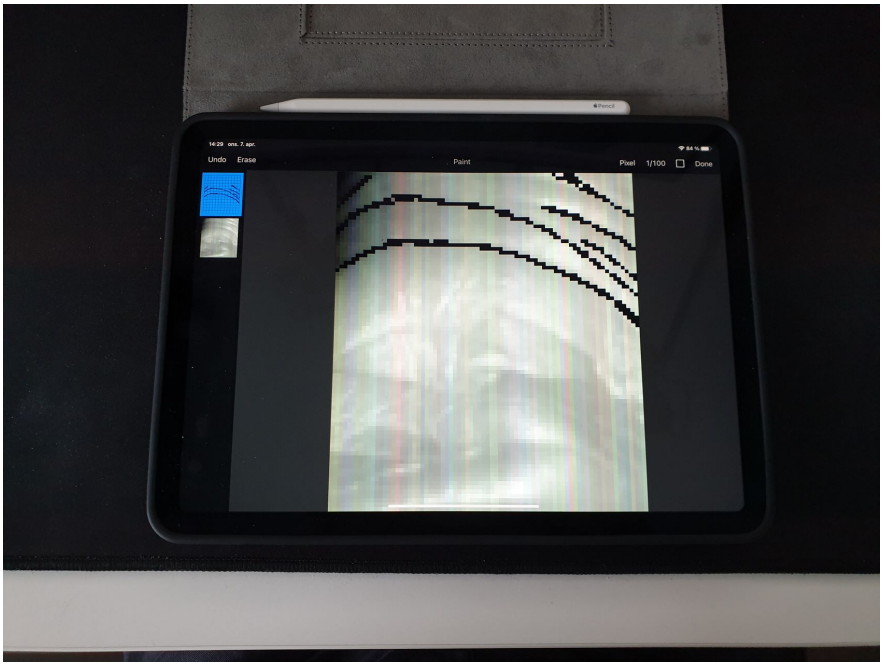

Figure 3.1.2: Manual annotation of the hyperspectral (HSI) dataset (appendix A) in Pixelmator. The annotation process of the HiRes dataset is equivalent.

Figure 3.1.3: The box used for obtaining homogeneous light condition and fixed distance and angle to the hand. Note that this image was taken months after data collection, and that at the time of data collection, the interior painting of the box was not damaged.

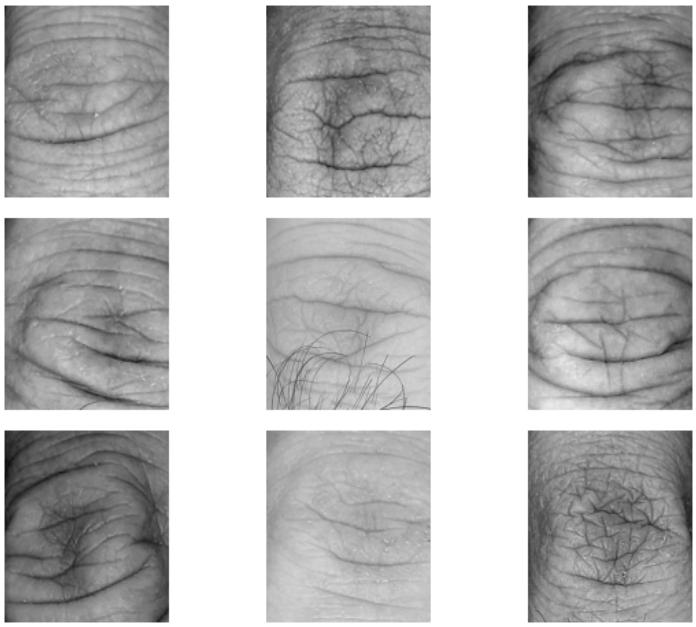

Figure 3.1.4: The high resolution dataset after grayscaling.

phone. The images captured were in 4:3 aspect ratio, 4032×3024 px and stored in JPEG format.

The images were then transferred to an Apple iPad Air 4, where the middle finger PIP joint was cropped. The software Pixelmator 2.6.3 [43] was used for manual annotation on an iPad, see fig. 3.1.2. A single image was loaded into the software. Then, a new layer was added. Using the `Paint/Pixel` tool, wrinkles were traced to the best of the author's abilities. When all the wrinkles were traced, the `Fill` tool was used to overlay each trace with white color. Then, a new black layer was added, and the white traces were moved to the front. An Apple Pencil 2 was used for drawing.

During annotation, one participant was disqualified since the close up image showed abnormally dry skin to the point in which it impaired accurate wrinkle annotation. Furthermore, in a retrospective review of annotation quality, two participants were disqualified since those two samples exhibited a lot of tiny wrinkles with low contrast levels, prohibiting accurate annotation. Thus, the final HiRes dataset consists of only 9 manually annotated images. The final mask layers had dimensions 414×483 px.

The HiRes dataset is shown in fig. 3.1.4.

Preprocessing and Data Augmentation

Grayscaling and contrast histogram equalization

In order to minimize the effect of different skin pigmentation levels in the algorithm performance, the HiRes dataset is subject to grayscaling by axis removal, leaving only the red color channel. We use the red color channel since it is less dependent on varying skin pigmentation (melanin), which strongly effects the reflectance spectrum for shorter wavelengths (e.g. blue and green, see section 2.1). This was also suggested by Cula et al. [16; 17]. Then, the images are subject to contrast histogram equalization. This method gives homogeneous contrast levels across skin pigmentation levels when the image is confined to a particular region [17].

Padding

Each input image is mirrored such that the resulting image width w and height h is divisible by 2^k where k in an integer. Given input dimensions of 414×483 px, the desired output dimensions are 512×512 px ($k = 9$). This is necessary in order for the U-Net to predict masks with the same dimensions as the input images, providing lossless[1] calculation of the Dice coefficient.

Smoothing the manual masks

From `Pixelmator` we get staircase shaped masks, especially visible in curves. To counteract on this effect, the masks are subjected to anti-aliasing by Gaussian blur ($s = 5$), thresholded (`threshold limit=128`), and transformed by a morphological closing (kernel size 5×5). This processing smooths the staircase shapes and makes natural connections between neighboring masks, yielding a better annotation by qualitative evaluation. The result is shown in fig. 3.1.5.

Data augmentation

The augmented dataset is created separately from the `PyTorch Dataloader` pipeline using `Albumentations` [27] and stored on the disk. We use the following transformations for data augmentation:

- Elastic transforms
- Grid distortions
- Optical distortions
- Grid distortion followed by optical distortion

[1] If input and output dimensions does not match, a resizing technique using methods such as nearest neighbors, bi-linear scaling etc. must be used, hence reallocating a possible loss from the interior of the U-Net algorithm, to the exterior.

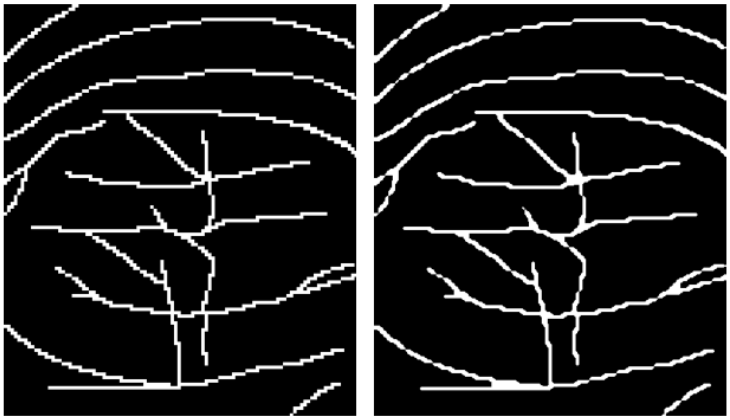

Figure 3.1.5: Left image shows the manual masks output from the annotation tool. The right image shows the manual masks smoothed by anti-aliasing, thresholding and morphological closing.

Table 3.1.1: Parameters for elastic transforms.

Parameter	Value
Alpha	120
Sigma	6
Alpha affine	3.6

In addition to the above given transformations, `horizontal flip` is applied to input images. Furthermore, each transformation is applied 10 times with a new random seed each time. The parameters for elastic transform and optical distortion are given in table 3.1.1 and table 3.1.2 respectively. No parameter input is given in the documentation [27] for grid distortion, hence the parameters are unknown and left as-is. However, fig. 2.3.7 shows a square lattice grid being transformed by the exact same grid transform as the one we use here.

The preprocessed input image and masks, as well as their augmentations are shown in fig. 3.1.6.

3.2 The Frangi-Gabor Process

The Frangi-Gabor process is a process in which we use methods from classical computer vision for wrinkle segmentation. The process was initially de-

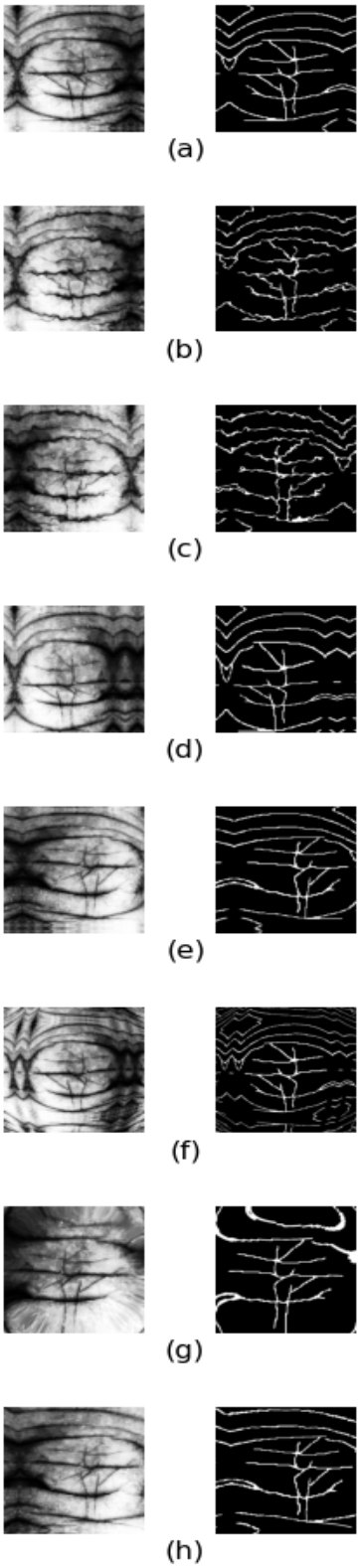

Figure 3.1.6: Data augmentation of input images (left) and their corresponding masks (right). (a) contrast histogram equalized and padded input image, (b) elastic transform, (c) flipped elastic transform with new seed, (d) grid distortion, (e) flipped grid distortion with new seed, (f) optical distortion, (g) flipped optical distortion with new seed, (h) grid distortion followed by optical distortion.

Table 3.1.2: Parameters for optical distortion.

Parameter	Value
Distort limit	2
Shift limit	0.5

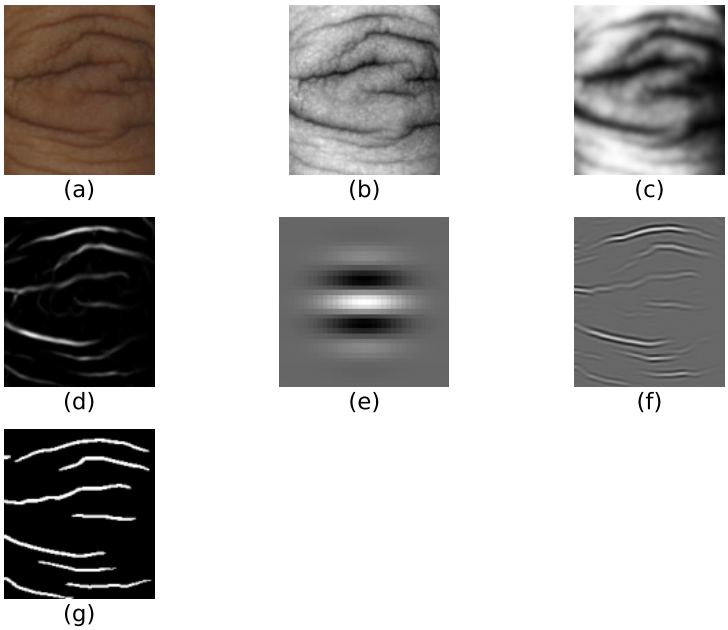

Figure 3.2.1: The Frangi-Gabor process including preprocessing stages. (a) Original input image. (b) Grayscaled by axis removal. (c) Contrast histogram equalization and Gaussian blur. (d) Frangi filtered (e) Gabor kernel with $\omega = 0.1$ and $\theta = \frac{\pi}{2}$. (f) Gabor filtering of the Frangi filtered image. (g) Frangi-Gabor masks after hysteresis thresholding.

veloped on the Kumar dataset during the pre-project. However, the current implementation is updated. This section explains the preprocessing, filtering and parametrization of the algorithms used. The full process is visualized in fig. 3.2.1. A sample from the dataset with wrinkle masks overlayed is shown in fig. 3.2.2.

3.2.1 Preprocessing

Grayscaling and contrast histogram equalization similar to that of section 3.1.2 are applied inline in the FG pipeline.

Figure 3.2.2: Original input image with the Frangi-Gabor masks overlayed.

3.2.2 Frangi Filtering (FRF)

The preprocessed images were then filtered with a Gaussian kernel of size $s = 3$. The output is shown in fig. 3.2.1(c). Gaussian filtering prior to the application of the FRF (section 2.3.1) is important as to avoid noise in the FRF output. The Gaussian filter effectively removes small textures, hair etc. Then the FRF was applied with $s_{FRF} = 1$. The value of $s_{FRF} = 1$, the size of the Gaussian kernel used in the filter, was chosen by qualitative assessment of the filter performance.

3.2.3 Gabor Filtering

The FRF is omni-directional, but we only want horizontal ridgelike structural features in our output, since vertical features are prone to be noise. By applying a Gabor filter (section 2.3.2) with $\omega = 0.1$ and $\theta = \frac{\pi}{2}$, we effectively remove vertical features. The frequency parameter ω can be viewed as a spatial size parameter. The value of ω was chosen based on qualitative evaluation. θ, the direction of the filter, is measured relative to the vertical axis of the image. Hence, for filtering horizontal structures, a value of $\theta = \frac{\pi}{2} + \pi k$ where k is

an integer, is required. After filtering, the remaining structures have a direction not smaller than approximately $\pm 50^{\circ}$ with respect to the vertical axis. ω impacts the slack around the directionality chosen by θ. In general, lower ω gives larger directional slack.

3.2.4 Hysteresis Thresholding

The final step of the Frangi-Gabor annotation process is the thresholding procedure. We employ hysteresis thresholding (section 2.3.3) with `high`=71 and `low`=5. The values refer to pixel intensity values. The quality of the output masks, and thus the tuning of the parameters were evaluated qualitatively.

3.3 The U-Net Process

The U-Net is a neural network architecture for semantic segmentation. It can be considered to be a CNN autoencoder. In our application, we want to train the network to learn a sparse representation of wrinkles.

In this section, we give the overall structure of the U-Net implementation, training setup and inference.

3.3.1 Implementation

The U-Net is implemented in Python using the `PyTorch` framework for deep learning based on [3; 44], with a slight modification to the originally published network architecture, as given in section 2.4.4. In our network, the double convolutions have `padding=1`, whereas the original architecture has `padding=0`.

3.3.2 Preprocessing and Data Augmentation

All preprocessing and data augmentation are done prior to training. This saves resources in terms of available computing power during training, compared to on-demand preprocessing in the training pipeline. The details of the preprocessing and data augmentation are given in section 3.1.2.

3.3.3 Training

We use the `Adam` optimizer with learning rate scheduler `ReduceLROnPlateau`. `ReduceLROnPlateau` is used to reduce the learning rate when the network parameters lies on a plateau in the loss landscape. This is because these points

might be saddle points, in which we do not wish to make large weight adjustments, since this can lead us to sub-optimal solutions. The loss function is binary cross-entropy (implemented with `BCEWithLogitsLoss`, which is more numerically stable compared to `BCELoss` with preceding sigmoid layer) since we only classify two classes; wrinkle and non-wrinkle. Training progression is recorded with `tensorboard` and performance is measured with the Dice coefficient.

3.3.4 Inference

Inference is the prediction of wrinkle masks on an input image. The U-Net model and the trained weights are loaded onto GPU memory. The input image is forward passed through the network, and the output is a binary greyscale image with segmentation masks.

3.4 U-Net Models

In this section, we present two models having equivalent hyperparameters and architecture, but they are trained and tested using different assessment strategies.

3.4.1 Model A: Supervised Learning on the HiRes Dataset with Train-Test Split Assessment Strategy

This model features supervised learning with manually annotated masks Y on the high resolution (HiRes) dataset X. Furthermore, it follows a typical train-test split assessment strategy.

Training

We use `Adam` optimizer with hyperparameters as given in table 3.4.1. The 9 sample dataset is data augmented into 639 samples. Then, a test set of 20% is uniformly sampled from the augmented samples, giving a total training set of 511 samples and test set of 128 samples. The train set is further split into a train and tuning set (10%), giving $n_{train} = 460$, $n_{tune} = 51$ and $n_{test} = 128$.

Estimating Generalized Predictive Accuracy

The generalization error E_{gen} can be estimated by computing the test error over the test set,

Table 3.4.1: Model A hyperparameters.

Hyperparameter	Value
Epochs	20
Batch size	2
Learning rate	0.0001

$$E_{gen} \approx E_{test} \tag{3.4.1}$$

In this project, we are more concerned about the generalized predictive accuracy. Using the Dice coefficient (DSC) as a measure of predictive accuracy, generalized predictive accuracy DSC_{gen} is estimated by computing the DSC over the test set,

$$DSC_{gen} \approx DSC_{test} \tag{3.4.2}$$

and similarily with the Jaccard Similarity index (JSI).

3.4.2 Model B: Supervised Learning on the HiRes Dataset with Leave-One-Out Cross-Validation Assessment Strategy

This model features supervised learning with manually annotated masks Y on the high resolution (HiRes) dataset X. Furthermore, it follows a Leave-One-Out Cross-Validation (LOOCV) assessment strategy.

Training

We use `Adam` optimizer with hyperparameters as given in table 3.4.2. The 9 sample dataset is divided into 9 folds; each training fold containing 8 samples and the corresponding test fold containing only one sample. Then, each training and validation fold are augmented separately, yielding 71 augmented samples for each original sample. Additionally, a 10% tuning set is extracted from each training fold for tracking training performance. Thus, each fold has $n_{train} = 512$, $n_{tune} = 56$ and $n_{val} = 71$.

Table 3.4.2: Model B hyperparameters.

Hyperparameter	Value
Epochs	20
Batch size	2
Learning rate	0.0001

Table 3.5.1: Hardware used for Frangi-Gabor annotation and U-Net training and inference.

Component	Type
Processor	AMD Ryzen 3700X
Memory	32GB DDR4 3600MHz
GPU	Nvidia GTX 1070 8GB
Storage	500 GB M.2 NVMe 2200/2000 MB/s R/W

Estimating Generalized Predictive Accuracy

The model is trained $k = 9$ times, each time with a new training fold. Using the Dice coefficient (DSC) as a measure of predictive accuracy, generalized predictive accuracy DSC_{gen} is estimated by cross-validation predictive accuracy DSC_{val}^{CV}, which is computed by averaging DSC_{val}^{i} over all training folds,

$$DSC_{gen} \approx DSC_{val}^{CV} = \frac{1}{k} \sum_{i=1}^{k} DSC_{val}^{i} \tag{3.4.3}$$

where DSC_{val}^{i} is the predictive accuracy over training fold i.

An equivalent estimation is carried our for the Jaccard Similarity Index (JSI).

3.5 Setup

This section lists the hardware and software used in the project.

3.5.1 Hardware

The hardware specification of the computer is shown in table 3.5.1. Photography equipment, and other equipment used is listed in table 3.5.2.

Table 3.5.2: Other hardware and equipment.

Component	Type
Camera	Samsung Galaxy S9, back-side camera
Light source	Celly Selfie Flash Light Pro White
Face masks	IIR (EN 14683:2019)
Nitrile gloves	PPE Cat. III (ISO 374-1:2016/Type B, ISO 374-5:2016)

Table 3.5.3: A subset of the software used in the project.

Software	Type/Version
Operating system	Ubuntu 20.04.1 LTS 64-bit, GNOME v3.36.3
Python	v3.8.5
PyTorch	v1.6.0
Anaconda	v4.8.3
matplotlib	v3.3.2
NumPy	v1.19.2
Pandas	v1.1.3
Pillow (PIL)	v8.0.0
scikit-learn	v0.23.2
SciPy	v1.5.2
torchvision	v0.7.0
OpenCV	v4.2.0
Spectral Python (SPy)	v0.21

3.5.2 Software

A subset of the software used, including the operating system, is given in table 3.5.3. Relevant drivers are given in table 3.5.4.

Table 3.5.4: Relevant drivers.

Drivers	Version
Nvidia driver	450.66
CUDA	v11.0
cuDNN	v7.6.5

3.5.3 COVID-19 Considerations on Data Collection

The collection of samples were performed in the same student housing where the author lives, on a “friends and family” basis, as to ensure that data collection did not increase the risk of COVID-19 infection among participants and the examiner. During examination and collection, both the participant and the examiner wore face masks. The examiner wore nitrile gloves. See table 3.5.2 for type details. The participants used an 83% alcoholic hand disinfectant prior and posterior to the examination and data collection. The number of eligible participants were limited by infection control restrictions.

3.5.4 Ethical Considerations

Participants were given written and spoken information on the project, how their data would be managed, and they were showed examples of how their data would be presented in the thesis.

All participants are above the age of 18, and enrolled in/or having finished higher education study programmes at the Norwegian University of Science and Technology (NTNU).

Together with the hand image, information about gender and year of birth were collected. Participants have the right to revoke their data consent at any time, without having to give any reason. Thus, in order to enable for deletion of data, the name of each participant was recorded and stored in a separate location from the raw data, according to applicable privacy regulation and laws.

Chapter 4

Results and Discussion

This chapter presents the results and discussion. First, results are presented in a point-wise manner for each model in section 4.1. In section 4.2, we discuss the performance and behaviour of each model, primarily focusing on the development cohort. Finally, section 4.2.3, take an outlook towards the development and assessment process for deep learning studies in the medical field.

4.1 Results

4.1.1 Frangi-Gabor Process

Predictive Accuracy and Qualitative Performance

The Frangi-Gabor (FG) process was applied to the HiRes dataset. Figure 4.1.1 shows the preprocessed input images in the left column, manual annotations in the middle column, and the right column shows the FG output.

The FG performance was quantitatively evaluated with the Dice coefficient (DSC) and Jaccard Similarity Index (JSI). The results are given in table 4.1.1.

Computational Performance

The FG annotation process, including loading of an RGB image, preprocessing, filtering, postprocessing and saving the mask as a PNG file takes approximately

Table 4.1.1: Frangi-Gabor performance on the HiRes dataset.

	DSC	**JSI**
Test	0.30	0.18

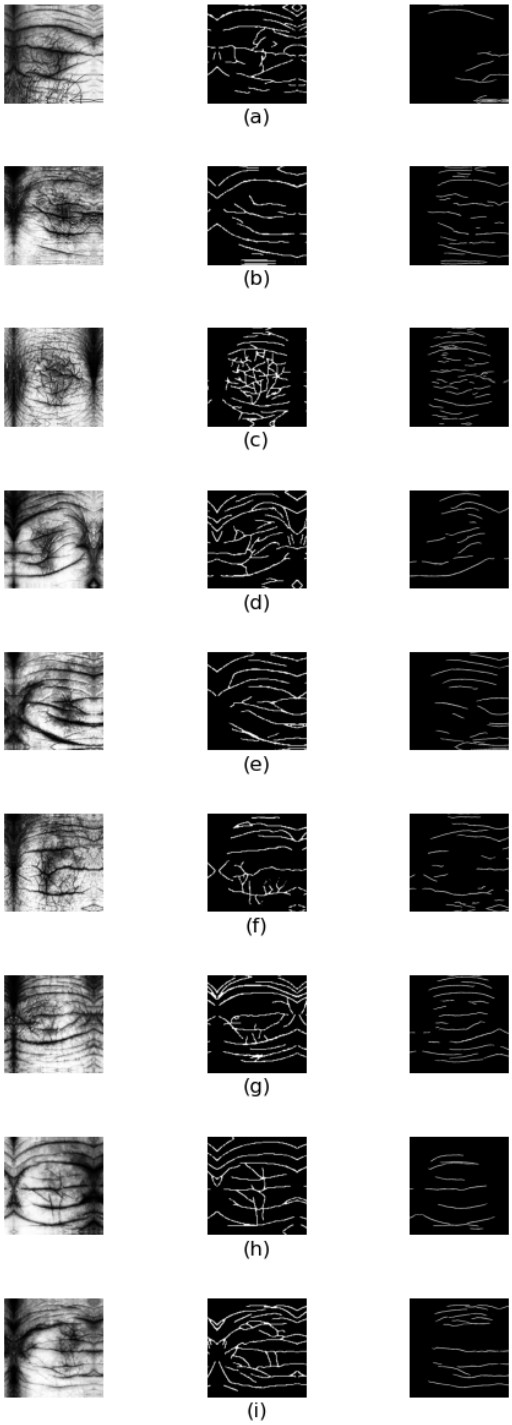

Figure 4.1.1: Frangi-Gabor process applied to the HiRes dataset. The left column shows the preprocessed input images, the middle column shows the manual annotations, and the right column shows the Frangi-Gabor annotations.

Table 4.1.2: Model A performance. JSI is not computed during evaluation of the tuning set since it is undefined for empty predictions, which may occur in early training steps.

	DSC	JSI
Tuning	0.81	–
Test	0.82	0.69

292 ms per image (3.42 images/second).

4.1.2 U-Net Model A

Predictive Accuracy and Qualitative Performance

Figure 4.1.2 shows Model A qualitative performance on the test set. The left column shows in preprocessed and data augmented input images, the middle column shows the manual annotation, and the right column shows Model A predicted masks. The predicted masks does indeed look very similar to those of the manual annotation.

DSC and JSI performance on the tuning and test set are given in table 4.1.2. Similarity metrics suggest good predictive accuracy of this model.

Computational Performance: Training

Figure 4.1.3 shows predictive accuracy (DSC) on the tuning set during training. Figure 4.1.4 shows cross-entropy loss during training. Training the model takes 29 minutes.

Computational Performance: Inference

Inference on each image takes about 65 ms after the dataset have been loaded to GPU memory. This is not including preprocessing time; preprocessing is applied in a separate process and takes about 85 ms per image. Thus, the total inference time can be seen as about 150 ms.

4.1.3 U-Net Model B

Predictive Accuracy and Qualitative Performance

Figure 4.1.5 shows Model B performance on each validation fold. The left column shows in preprocessed and data augmented input images, the middle column shows the manual annotation, and the right column shows predicted

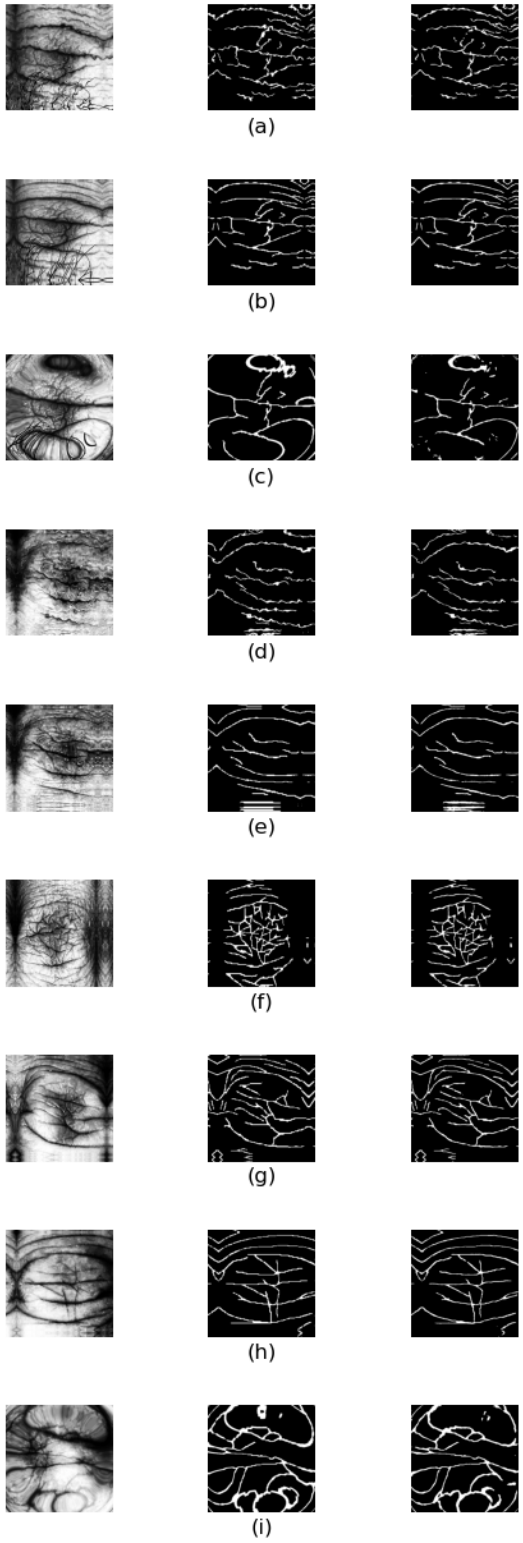

Figure 4.1.2: Model A test performance. The left column shows in preprocessed and data augmented input images, the middle column shows the manual annotation, and the right column shows predicted masks.

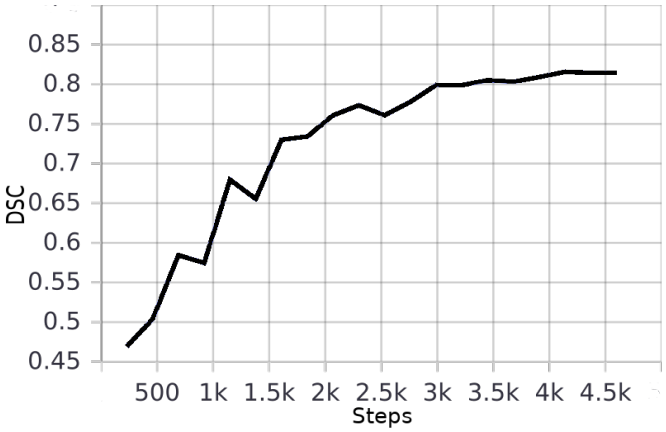

Figure 4.1.3: Model A predictive accuracy (DSC) on the tuning set during training.

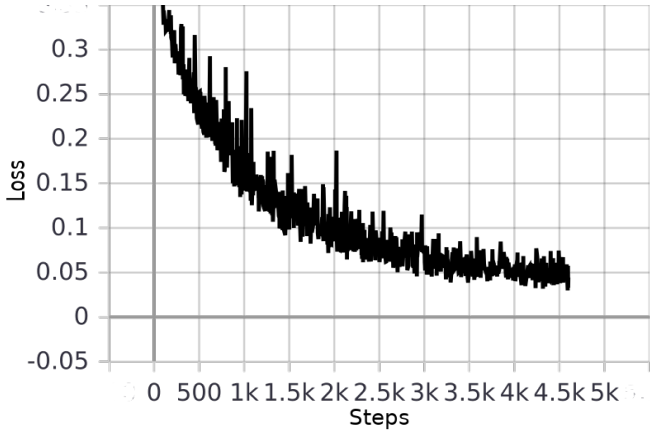

Figure 4.1.4: Model A cross-entropy loss during training.

masks. DSC and JSI on each validation fold are given in table 4.1.3, where the test score is estimated by averaging the validation score from each fold.

Computational Performance: Training

Figure 4.1.3 shows predictive accuracy on the tuning set during training. Figure 4.1.4 shows cross-entropy loss during training. Each fold is denoted by a separate color.

Training time of each fold is approximately 31 minutes. Hence, the total training time for all $k = 9$ folds is about 4 hours and 39 minutes. Additionally, validation of each fold takes about 5 seconds (15.39 images/second, 65 ms/image).

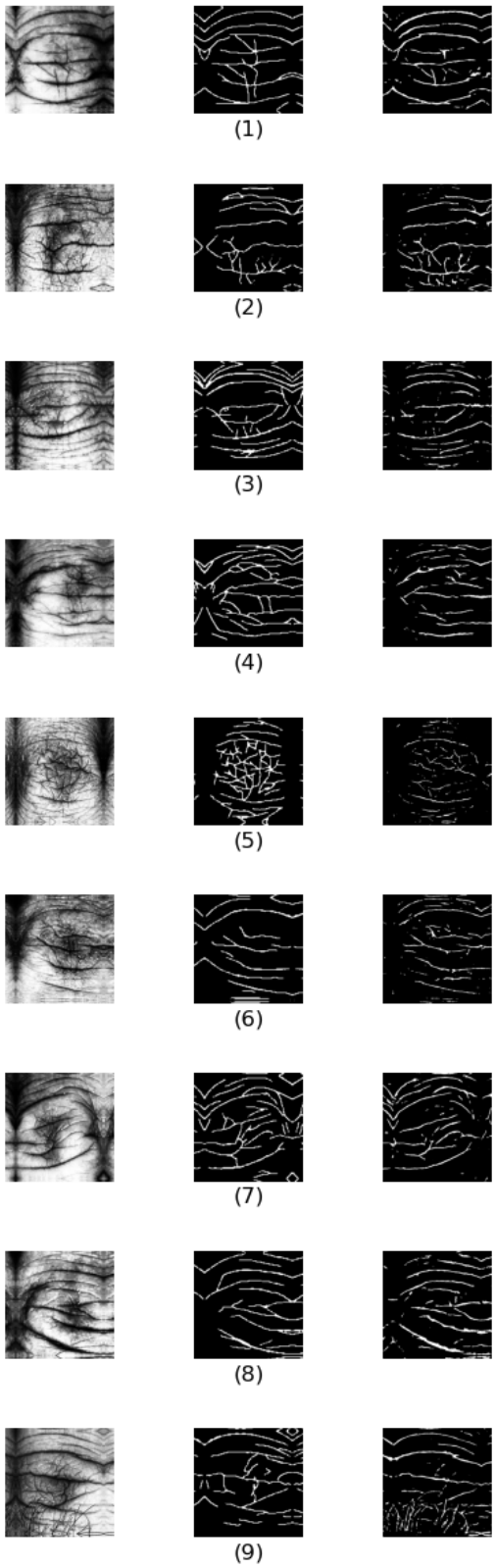

Figure 4.1.5: Model B test performance. The left column shows in preprocessed and data augmented input images, the middle column shows the manual annotation, and the right column shows predicted masks. Each row corresponds to the training/validation fold indicated numerically.

Table 4.1.3: Model B performance. Since Model B does not have a separate test set, test performance is estimated by the mean μ of all validation scores and standard deviation σ.

	DSC	**JSI**
Validation 1	0.50	0.33
Validation 2	0.49	0.33
Validation 3	0.41	0.26
Validation 4	0.43	0.27
Validation 5	0.32	0.19
Validation 6	0.42	0.26
Validation 7	0.51	0.34
Validation 8	0.50	0.33
Validation 9	0.34	0.23
Test $\mu \pm \sigma$	$\mathbf{0.44 \pm 0.07}$	$\mathbf{0.28 \pm 0.05}$

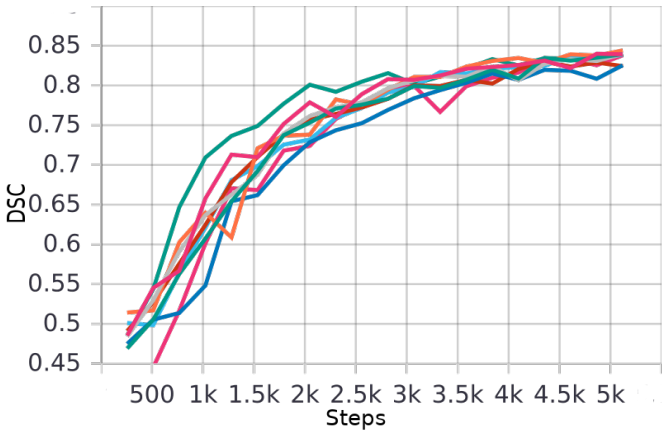

Figure 4.1.6: Model B predictive accuracy on the tuning set during training, each fold denoted by separate colors.

Computational Performance: Inference

Inference on each image takes about 65 ms after the dataset have been loaded to GPU memory. This is not including preprocessing time; preprocessing is applied in a separate process and takes about 85 ms per image. Thus, the total inference time can be seen as about 150 ms.

4.2 Discussion

In this section, we will first discuss the performance of the Frangi-Gabor (FG) process based on the results in the previous section. This will give motivation

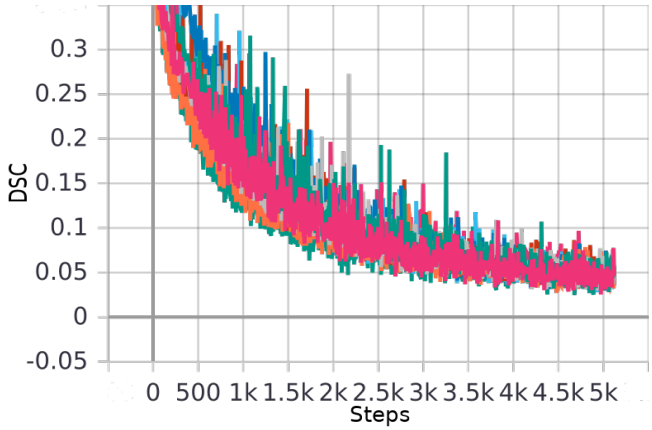

Figure 4.1.7: Model B cross-entropy loss during training, each fold denoted by a separate color.

for why one would consider migration to a ML method. Then, Model A is introduced, first with a discussion regarding its performance as we see it with its current training and assessment strategy; then, we will highlight problems and suspicions with the model, which leads us over to Model B. Model B is then presented, first with a discussion on performance, and then compared to Model A and the FG. Moreover, we discuss cross-validation compared to TTS, as well as other important aspects of supervised learning, such as confidence in ground truth and the manual annotation process. Finally, we take on a broader view on the development and assessment process of deep learning models for medical applications.

4.2.1 Frangi-Gabor Process

FG delivered a predictive accuracy of DSC=0.30/JSI=0.18 on the HiRes dataset. The similarity measures are not very high, suggesting that the algorithm is strongly under-performing wrt. to the manual annotations. Consider fig. 4.2.1, where manually annotated masks are labeled blue, correct predictions from FG are labeled green and mis-predictions from FG are labeled red. Note that the FG masks are slightly shifted upwards from the manual masks. Also notice the staircase shape of the manual masks, where the FG predicts a smooth curve. This is especially visible in the lower right annotations. Both of these examples decrease DSC/JSI. Nevertheless, FG is still only able to make predictions on horizontal ($\pm 50^\circ$) wrinkles, thus it will always under-perform to a certain degree compared to the manual annotations.

FG is a process consisting of a pipeline with several well-defined steps, such as color and contrast modifications, anti-aliasing, various filters and thresholding techniques. Having a well-defined model such as this is good for inter-

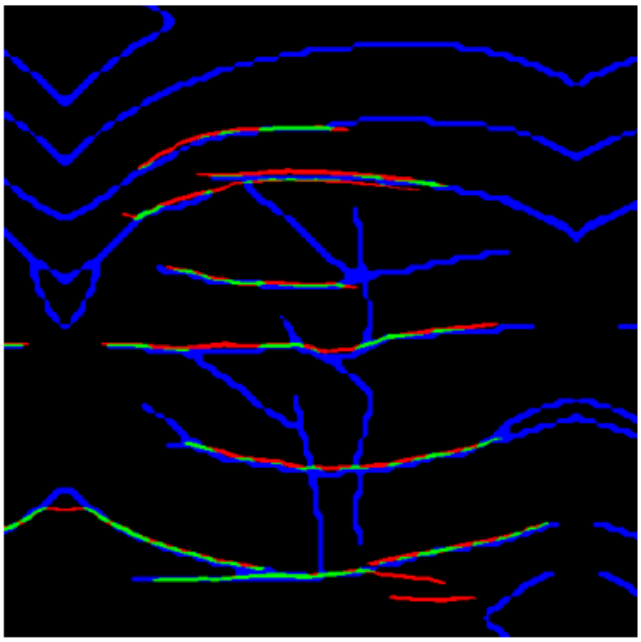

Figure 4.2.1: FG masks compared to manual masks. Manual mask are blue, FG masks overlapping with manual masks are green, and FG masks non-overlapping with manual masks are red.

pretability: we know that the algorithm finds wrinkles by probing a second-order derivative of a Gaussian kernel of size s, which we call a *probing kernel*, over the image. If we change input image dimensions, we can adjust the probing kernel size s and the Gabor frequency ω to fit better with new spatial sizes of wrinkles. Over- and under-estimation of wrinkles can be adjusted by tuning the hysteresis threshold parameters. Even the directionality of the algorithm can be changed by adjusting θ.

A drawback of the FG process, is that due to its transparency and interpretability, the behaviour of the algorithm is very strict and limiting, imposing rather hard cut-offs on spatial sizes and direction of wrinkles. By e.g. removing the Gabor filter, thus allowing for omni-directional predictions, we were not able to find parameters that gave any decent predictions on the Kumar dataset at all. Slight modifications to the hysteresis threshold limits also significantly worsens predictions.

It is probably possible to increase FG performance. For example, we informally

note that we believe that introducing a contrast histogram equalization step between Gabor filtering and hysteresis thresholding would increase performance by extending and strengthening the dark ridges (section 2.3.3) produced by the Gabor filter, thus enabling for relaxation of the hysteresis parameters.

In terms of computational performance, the FG delivers acceptable results. Keep in mind that the FG implementation in this project is on a prototype level, hence it is probably possible to increase computational speed further. From the pre-project to the current implementation, computational time per image was shortened by 40%[1] just by optimizing how the algorithm loads data in the pipeline. With the current speed of 292 ms per image in the HiRes dataset, the process is fast when considering one by one predictions, however, FG delivers about twice the inference time compared to the U-Net models. Anyway, the speed offered in the current implementation would be acceptable for one-by-one analysis in a clinical setting.

Looking towards a more flexible model

The strictness of the FG process, as well as the difficulties involved with method development, motivated our search towards a more flexible and easy-to-use model. A natural step was looking towards deep learning based computer vision models, which have shown remarkable results within a range of applications in the past recent years [10]. The U-Net is a well-known model for semantic segmentation. It won the 2015 ISBI cell tracking challenge, while being trained on only 30 samples of original data (which was later augmented) of the cell membrane of the Drosophila ventral nerve cord (VNC). In the microscopy images, the cell membrane have some similarities to wrinkles. Thus, trying out the U-Net for our wrinkle segmentation problems seemed to be worth a try.

4.2.2 U-Net Models

Model A

Model A was trained on a strongly augmented dataset, using a typical train-test split (TTS) training and assessment strategy. Each of the 9 samples from the original dataset were augmented into 70 transformed samples, giving a total dataset size of $n^A_{dataset} = 639$, including the original samples. Withholding 20% for the test set, and 10% of the remaining data for tuning, gave $n^A_{train} = 460$, $n^A_{tune} = 51$ and $n^A_{test} = 128$.

By qualitative evaluation of the predicted masks compared to the manual masks in fig. 4.1.2, it looks like the model works really well. We see that the predicted

[1] A reduction from 100 ms to 60 ms inference time on the Kumar dataset.

masks are similar to the manual masks both in the general structure of wrinkles, but also in the details. This is backed up by relatively high similarity scores (DSC=0.82, JSI=0.69) on the test set as given in table 4.1.2. Also, the tuning score and test score of Model A are in close proximity (DSC=0.01 difference). These test scores are comparable to earlier results on wrinkle detection, where test results typically lie in the JSI=60-90% range [1; 16; 18; 17]. However, the Cula et al. method performed in the sub-20% JSI range when evaluated on the Ng et al. setup [17]. Cula et al. emphasized the importance of high quality input to their algorithm; they used a polarized light imaging system to separate between skin surface spatial geometric structures and subsurface skin features like color variations. Thus, the employment of the Cula et al. method on images not captured and preprocessed in the same or a comparable manner seems futile. Furthermore, different annotation methods impacts the accuracy and behaviour of the resulting model [30]. We note that even though methods are presented to achieve relatively high performance, the score can change drastically when evaluated on other setups. Hence, the performance of existing methods are uncertain since they have not been evaluated using the same setup and data.

With the uncertainty regarding typical performance of wrinkle segmentation methods in mind, it is important to investigate on the high test scores of Model A further. First, we must dissect the properties of the data we are working on, and the handling of the data during development.

Limitations on the development cohort

The development cohort was sampled from a young, healthy, wealthy, highly educated and primarily light skinned population. We included methods aimed at mitigating the effects of different skin pigmentation; The combination of using only the red color channel, since it is the least affected by melanin (section 2.1), as well as contrast histogram equalization, which is effective at achieving homogeneous contrast levels across skin pigmentation levels when the image is confined to a particular region [17], was qualitatively evaluated on the Kumar dataset to be effective. Nevertheless, it is important to note that no quantitative tests have been conducted to verify this claim. Furthermore, by manual inspection we see that most of the participants have the same *macro-structures*[2] of the wrinkles: relatively long valleys, loosely forming an elliptic shape around the joint, with few major vertical wrinkles. Looking at the 503 sample Kumar dataset [42], it does not take long time before we realize that there are other typical macro-structures which we do not have in our dataset. There are also

[2]**Macro-structures** refers to the major visual characteristics of the wrinkle sample, e.g. if wrinkles are fine-masked, long valleys etc.

probably a vast range of variation in *micro-structures*[3] that we have not captured to a sufficient extent. The fact that the PIP/DIP dorsal finger skin has been researched in conjunction with biometrics [42], is a statement in itself on the individual uniqueness, and hence the variety across individuals in the data we are using.

Handling of data during model development

The HiRes dataset was augmented into 639 samples, which was further split into a train, tuning and test set, by drawing randomly with a uniform distribution. The model was trained on the train set, hyperparameters were tuned on the tuning set, and finally, the model was evaluated on the test set. This may have introduced a data leakage between the three sets. Since the data was augmented before splitting into separate sets, in particular before setting aside the test set, we have ended up in the situation where the train, tuning and test set all contain samples which *originates* from one another. The significance, and potential negative impacts from this data leakage, depend on the quality of data augmentation. Successful data augmentation by applying smaller transformations on the inputs while maintaining their relationship with the target output can increase generalization [8]. However, if the augmentations are too insignificant, the augmented data is essentially the same as the original data, yielding data leakage. Opposite, if the augmentations are too strong, the relationship between inputs and target outputs are occluded, thus resulting in bad performance [8].

In order to evaluate the quality of data augmentation, hereby the amount of potential data leakage, and subsequently uncover more knowledge concerning the estimated generalized performance of the model, we need to look for another training and assessment strategy.

In this new strategy, we impose that the train and test set must be completely disjoint. That is, they need to be disjoint also wrt. the origin of the data, not only individual (possibly augmented) samples. In Model A, we used 20% of the available data for testing. If we want to have a test set of approximately the same size while splitting prior to augmentation, this would require us to extract two samples (giving a 22% test set). But how to select which samples to use for testing? One way could be to randomly draw samples using a uniform distribution. Indeed, for a sufficiently large dataset, this would give us an unbiased selection of samples in the test set. However, it is crucial that the test set resembles the distribution of the population on which we want to obtain

[3] **Micro-structures** refers to the small local variations around wrinkles, that are not easily visible for human interpretation. It could be variations in the ridge-valley formations, whether the ridges are smooth or uneven in the longitudinal direction, etc.

a performance estimate. Extracting only two samples from a total of 9 available samples, is probably not a good strategy. For example, what would happen if we choose sample (5) and (9) (see fig. 4.1.5) for our test set – two samples that differ significantly from the other samples? It would probably give an under-estimation of model performance on the development cohort. Conversely, if we choose two of the other samples, which features more frequent macro-structures, we might over-estimate model performance.

Another important problem arising when extracting 22% of such a small dataset for testing, is that it takes away data containing features that are not present in other samples, which could have been used for estimating model parameters. Hence, removing the data from the training algorithm unables the model to adjust for the features found in these samples, effectively decreasing actual model performance.

We are in the crossroads between achieving maximum model performance (having a large training set), and achieving the best possible estimate of model performance (having a large test set).

Squeezing the data almost dry

A solution is changing from a TTS training and assessment strategy, to a cross-validation strategy. Many variations of cross-validation techniques exist [45], however we will be concerned with the k-fold cross-validation.

The first task is to find an appropriate k. From section 2.5.4 we know that the expected generalization error is decreasing as the training set increases. The expected generalization error decreases towards the *Bayes error* in an asymptotic fashion, or until the model saturates, whichever comes first [33]. This is the minimum expected error we can achieve with our model and dataset. For a sufficiently large dataset, we might reach minimum expected error before we have used the entire dataset; for example, for a certain dataset we might only need 80% of the data for training in order to obtain optimal expected performance. On small datasets however, we might find ourselves in a situation where the expected generalization error is ever decreasing for every new sample we include in the training set. In this case, we need to include as many samples as possible.

A special type of k-fold cross-validation, where $k = n$ (n being the size of the entire dataset) is called Leave-One-Out Cross-Validation (LOOCV). This method minimizes expected generalization error wrt. fig. 2.5.2, as Mosteller and Tukey comments: "*Suppose that we set aside one individual case, optimize for what is left, then test on the set-aside case. Repeating this for every case squeezes the data almost dry. (...) When practical, this approach is attractive*" [46; 38]. An empirical study by Martens and Dardenne [40] investigated different methods of

using small datasets in data-driven models, where small datasets were simulated by randomly extracting 40–120 samples from a true dataset of size 922. The extracted data were used for both training, and estimating predictive precision, while the remaining data (802–882 samples) were used as a control test set, to evaluate the estimates from the extract. The results showed that TTS was wasteful in that the true predictive performance was much lower, and showed greater variability, than when LOOCV was used. Furthermore, it was found that estimated generalization error of LOOCV was only slightly optimistic.

Model B

Model B is essentially the same CNN autoencoder as that of Model A. It has the same hyperparameters and network architecture. The only difference between the two models are the training and assessment strategies employed, where Model B uses LOOCV instead of TTS.

In Model B, each fold were augmented independently and separately from one another, resulting in 9 folds where each fold contained $n^B_{train} = 512$, $n^B_{tune} = 56$ and $n^B_{val} = 71$. Then the model was trained and evaluated on each validation fold $k = 9$ times, once for each of the original data samples. The tuning set n^B_{tune} was used to track performance during training.

Major wrinkles are in most cases successfully predicted by the model. By comparing manual masks and predicted masks, we are generally able to see that they are indeed masks belonging to the same input sample. However, the model is having difficulties in predicting special cases; consider validation fold (5) and (9). In validation fold (5), the model is asked to make predictions on an image which is very different from what it has seen before (note the fine-masked wrinkles). From the Kumar dataset, we know that this is a common type of macrostructures, however the HiRes dataset only contains one such sample. Thus, this sample has a completely different distribution compared to the other samples in the dataset, significantly decreasing predictive accuracy. Validation fold (5) has the lowest similarity measure (DSC=0.32, JSI=0.19) across all folds. Validation fold (9) is also very different from the other samples, as this sample contains dark hair. In this case, the model mis-classifies hair as wrinkles. Furthermore, it is unable to give good wrinkle predictions for the horizontal wrinkles located underneath the hair. Suddenly having hair in the validation sample is a broad change from the corresponding training fold, leaving the training fold and validation fold with unequal distributions. This validation fold has the second lowest similarity score (DSC=0.34, JSI=0.23) across all folds.

Note that Model A, which has images with hair in both the training and test set, does not mis-classify significantly on hair.

The generalized predictive accuracy of Model B was estimated by taking the av-

erage of the predictive accuracy for each validation fold DSC^i_{val}, giving $DSC_{gen,B} = 0.44 \pm 0.07$ ($\mu \pm \sigma$), see table 4.1.3. We immediately make three remarks; (i) the estimated performance is much lower than that of Model A, with an estimated reduction of -0.38 ± 0.07 DSC, (ii) the estimated performance is generally low, and (iii) there is relatively large variability in the estimated performance of each fold ($\sigma = 0.07$).

(i) by cross-validation, estimated predictive accuracy drops by more than 46% compared to TTS. As expected, we see that the cross-validated results are less optimistic. This aligns well with the conclusions from Martens and Dardenne [40], and the reflections from Mosteller, Tukey and Stone [46; 38]. However, the enormous drop in predictive accuracy cannot be explained solely by the introduction of cross-validation. A more likely reason for this result is that the data leakage between the train and test set that we previously noted in Model A, is indeed significant. This shows that the data augmentation we performed was insufficient in adequately introducing new variance in the dataset, resulting in Model A being strongly overfitted. The result also highlights why the test set should be *completely disjoint* from the training set, also in terms of the *origins* of the data.

The LOOCV trains the model on every available data sample while validating the model on data completely disjoint from training data. Hence, this method optimizes expected generalized performance, while still enabling us to give an unbiased estimate of generalized performance. The estimate is unbiased as long a the model is never adjusted based on validation scores.

The second remark (ii), the fact that Model B is not a very accurate model, is not very surprising. High capacity models trained on small datasets are prone to overfitting, which in turn yields bad generalized performance [33; 8]. Increasing the training data however, typically enables the model to find more general patterns. In case more training data is not possible to obtain, other methods exist to help the model generalize better. Such methods are called regularization techniques (section 2.4.1). Neither of the two U-Net models have embedded regularization techniques that apply to weights (e.g. dropout, $\mathcal{L}_p$-regularization, etc). We see from the results that data augmentation as a regularization technique alone was not sufficient. Remark (iii) on the variability shown in the validation scores, with scores in range $[0.32, 0.51]$ and $\sigma = 0.07$, further implies that the dataset is indeed too small, otherwise we would expect less variation across validation folds. Thus, in this situation, we believe that it is absolutely necessary to include a larger dataset to capture more natural variation.

The amount of data required to produce a good model for wrinkle segmentation is not easy to predict. We previously commented on the vast variation in demographics, macro- and micro-structures in the true population, hence the

number of features involved with wrinkle segmentation are large. Therefore, accounting for the *curse of dimensionality* suggests that we need a rather large amount of data in order to produce a viable model.

Interpretability vs. efficacy

It is worthwhile to make a quick comparison between Model B and the Frangi-Gabor (FG) process. Model B achieves 0.14 ± 0.07 DSC better estimated predictive accuracy compared to FG. Qualitative results shown in figures 4.1.1 and 4.1.5 show that Model B predictions give a more accurate reproduction of the true wrinkle masks, compared to FG. Two important characteristics of the FG process are responsible for this: (i) it is predicting wrinkles mostly in the horizontal direction ($\pm 50^{\circ}$) and (ii) it can only predict major wrinkles. Model B is omni-directional, and has no hard cut-off with regards to spatial wrinkle size, or directions. Immediately, these results show the motivation and temptation for migrating from a classical approach to a ML approach. However, even though we have employed cross-validation to minimize generalization error, and we have no data leakage, the small development cohort used in Model B leaves us critical and inconclusive wrt. performance on new unseen data. This suspicion is increased due to relatively high variability of the performance across validation folds as previously noted. An advantage with the FG compared to Model B, is that we actually know how it operates, and we know that it works provided correct inputs. Considering its embedded limitations, the FG performed well on the Kumar test set (88 samples), as shown in fig. B.0.1 (b) in appendix B. Thus, both increased interpretability, and the more extensive testing, give us increased confidence regarding the operation of the FG on unseen data.

Confidence in the ground truth

An important part in any supervised learning project, is the confidence in the ground truth, which in our case amounts to the quality of the manual masks Y. As far as the algorithm is concerned, Y is the ground truth, assuming that every pixel in the mask is perfectly annotated. However, this is seldom true [47; 30]. Human annotators, as well as algorithmic annotators, make errors. Thus, we must assume a confidence level below 100% on the proposed ground truth masks Y. The level of ground truth confidence is important in the evaluation of model performance; e.g. if the ground truth confidence is just 95%, then we can never claim that the model performs better than this, even if it performs with a 100% accuracy upon testing. Moreover, the mask quality have implications on the actual model performance.

Estimating ground truth confidence is often a matter of calculating the agreement of ground truth across annotators. Specifically, inter- and intra-reproducibility

are metrics of interest [30]. The *inter-reproducibility* is a measure of the agreement between different annotators. Ng et al. [17] (HHF) calculated the average inter-reproducibility between three programmers annotating the center-line of wrinkles in the Bosphorus dataset [48] to be 0.93 JSI. In a later article also by Ng et al. [1], proposing the Hessian Line Tracking (HLT) for wrinkle detection, the same annotation process was carried out, however this time, two of the annotators were computer scientists and the third annotator was an "experienced beauty therapist". In this trial, the inter-reproducibility between annotators was 0.94 JSI. Cula et al. found the inter-reproducibility between annotators to be above 88% [16]. In our project, there is only one annotator, thus no estimate on the inter-reproducibility can be calculated.

The *intra-reproducibility* measures the agreement between multiple annotations performed by the same annotator. Ng et al. [1] estimated the intra-reproducibility by letting the annotators first annotate the whole dataset on day one. Then the next day, they annotated 30% of the dataset again. The average intra-reproducibility was calculated to be 0.79 JSI.

The intra-reproducibility on our dataset was calculated to be 0.38 JSI/0.55 DSC by re-annotating 1/3 of the dataset two months after the first annotation session. This is significantly lower than that of the previously noted intra-reproducibility results. It is not straightforward to investigate the discrepancy between our results, and the results of Ng et al., since the articles we are comparing to do not provide full details on the annotation process. Nevertheless, we will comment on our results. Consider fig. 4.2.2. We see that the main structures of the wrinkles are commonly annotated across annotation sessions, however both sessions include details which are not present in the other. Also note the staircase-shaped curves, where discrepancy increases between sessions. Finally, we note that in some cases, annotations are separated by only one pixel width, thus the annotator was able to identify the wrinkles, however the low precision of the annotation tool impacted the ability to create overlapping segments.

The estimates of inter- and intra-reproducibility illustrate that manual wrinkle annotation is a difficult perceptual task with low consistency within the same annotator. This was also noted by [1; 17]. Eramian et al. [30] investigates different types of annotation methods (point-wise, stroke-wise etc.) in semi-automatic segmentation algorithms[4]. While the focus of the article is not directly relevant to our problem in particular, they provide some general remarks that we would like to reproduce:

- The type of annotation method impacts the accuracy of the model.

[4]Segmentation algorithms in which an annotator provides loose annotations (e.g. just marking a single point on an object), and then the algorithm annotates the rest.

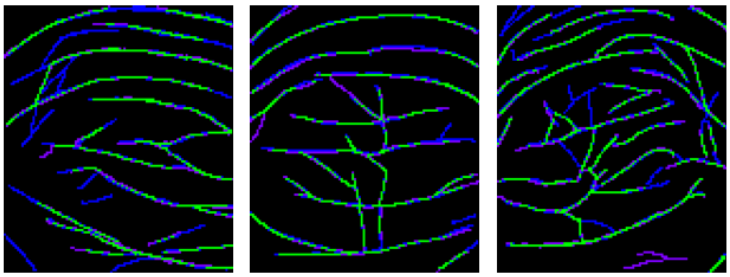

Figure 4.2.2: Intra-reproducibility on the HiRes dataset. Original masks on which the models was train on (prior to post-processing) are shown in blue, re-annotated masks for metric computation are shown in violet, and overlapping masks from both annotation sessions are shown in green.

- Time pressure impacts the workload[5] for annotators.
- The annotator experiences more mental and physical demand, and frustration when asked to use a more detailed annotation type (e.g. changing from point-wise to stroke-wise annotation). The annotator also puts in more effort in the latter annotation type.
- Annotator fatigue is cumulative over time and over annotation samples.
- The quality of the images themselves will potentially have an unintended impact on the performance of annotators, and subsequently the performance of the trained model.
- Images with objects to be segmented that are particularly dendritic (shapes with thin protrusions) will lead to targeting problems.
- Various contrast levels may impact the annotators' ability to correctly identify regions of interest.
- Images with a larger number of regions of interest ("more objects") increases the amount of wrong annotations.

Several of the above remarks are relevant in terms of manual wrinkle anno-

[5]Eramian uses the term **workload** as an aggregated measure of annotator confidence, fatigue, throughput and DSC accuracy.

tation both in our project. We use an annotation method where we annotate pixel-wise the valleys ridge-to-ridge. The fact that our annotations were performed under time pressure (see Preface), may have lead to high experienced workload. We can confirm from our experience during the annotation process that it was indeed frustrating and demanding. Lack of experience with the annotation tool lead to more frustration, and the annotator experienced significant neck pain during the annotation of the final samples. The images were not preprocessed before annotation. Low contrast levels thus resulted in perceptual difficulties in the annotation process; the annotator recalls that there was a feeling that the "wrinkles disappeared in front of my eyes" during annotation immediately after beginning to annotate a new wrinkle. When the masks were drawn, the annotator experienced that "the brain lost its ability to contextually perceive the continued location of the wrinkles". The annotation tool (`Pixelmator`) was unable to create smooth curves – instead it created staircase shaped curves (see fig. 3.1.5). In terms of achievable annotation accuracy, this is obviously negative. But this may also have a negative impact on annotator performance, as the confidence in the annotations decrease as the tool unables for precise annotation. Over the course of several annotations, this might have the undesirable effect in the annotator starting to put in less effort in being precise, since the tool anyway inflicts low precision, thus resulting in a double-negative impact on the overall annotation quality, and subsequently on model performance.

Mask quality impacts on model performance

With intra-reproducibility of only 0.38 JSI/0.55 DSC on our dataset, confidence in the ground truth is very low. Essentially, this means that the annotator does not agree with himself in the exact location of wrinkles. This uncertainty propagates into the algorithm during training, since the apparent features denoting a wrinkle is not very definite.

Post-processing of manual masks

Due to the difficulties involved with manual annotation, and especially the staircase-shaped curves imposed by the annotation tool, we performed post-processing of the manual masks in order to smooth the curves. Qualitatively evaluated, the process involving anti-aliasing, thresholding and morphological closings were effective at smoothing the curves, however the processed curves were sometimes still slightly staircase-shaped. More aggressive processing was not possible, since this resulted in excessive amounts of large clusters in wrinkle junctions, due to the morphological closing's re-joining characteristics on disjoint structures in close proximity.

Computational feasibility

Training Model A takes about 29 minutes, while each training fold of Model B takes about 31 minutes. The latter model uses 11% more samples in each training cycle, with increased computational time of about 7%. However, Model B is trained 9 times, giving a total training time of 4 h 39 m. Both models train relatively quick compared to the original U-Net publication, where the model was trained for 10 hours [3]. Ronneberger et al. [3] did not use cross-validation, thus comparing Model A with the original U-Net shows that the former trains in about 1/20 of the time of the original network. The reason is that Model A saturates quicker since the amount of data is smaller and more noisy. Ronneberger et al. had 30 original samples which was augmented using some of the same techniques as those of ours. Furthermore, their dataset consisting of professionally annotated images of the Drosophila ventral nerve cord (VNC) features more accurate masks.

Inference time of 150 ms on each image for both Model A and B is acceptable for use in a clinical setting where one by one analysis is performed. The U-Net models are about twice as fast as FG, which has an inference time of 292 ms per image.

4.2.3 Deep Learning Studies in Medicine

In this section, we look up and take on a broader view on the development and assessment process of deep learning models for medical applications. We view some of our findings in light of recent approaches aimed at achieving more stringent development and assessment practices for deep learning studies in medical research. The discussion is also relevant for non-medical applications.

Designing deep learning studies for medical research

For medical research, the structure and implementation of randomized controlled trials (RCT) are well established. RCTs are aimed at reducing bias in experimental trials. Eligible participants are randomly allocated to groups, typically one group receiving some form of new treatment, and one group being the *control group*, receiving traditional treatment or placebo [49]. The trials are typically conducted in a *blinded* fashion [50]; in a single-blinded trial, the participants do not know whether or not they receive the actual new treatment or not. It is used to reduce the risk of errors, as participants can produce false results if they know they receive actual treatment. In a double-blinded trial, neither the participants nor the clinicians know who are under (new) treatment, and who are receiving standard treatment or placebo. This method is

stronger than a single-blinded trial, since it removes the observer bias, that is, conscious and unconscious bias the clinician might impose on the results based on their own beliefs in the treatment on trial. Interventional[6] RCTs must be prospectively registered [51; 52], giving information on ethical considerations, analysis procedure, number of participants, expected study power and significance, etc. Observational studies does not require registration, but it is highly recommended by best practice. Among the benefits of prospective registration and results submission are increased motivation to fulfill ethical obligations, reduce publication bias[7], more efficient allocation of research funds, public records of basic study results in a standardized format, thus facilitating systematic reviews.

Many ML studies in medicine report performance comparable or better than clinicians [8], however many of them were found to be at high risk of bias and deviated from existing reporting standards [9; 8], resulting in potentially misleading results and claims [9; 54]. While the use of RCTs are readily used in medicine, no such methods exist specifically for ML and deep learning (DL) studies. In particular, frequent lack of evaluation on external data (see section 2.5.3) as well as development on too narrow datasets limits the medical utility for many of the methods presented in research [8]. While RCTs can very well be used for assessing safety and efficacy of DL systems [54], the fact that many DL studies are retrospective and observational promotes the need for custom protocols more adjusted to the needs for DL research on medical applications [8].

Kleppe et al. [8] proposes a (non-exhaustive) list of items considered suitable for such protocols. The protocol, termed *Protocol Items for External Cohort Evaluation of a deep learning System (PIECES)* (see [8], BOX 3) is inspired by the RCT structure, but aims at also capturing the retrospective and observational characteristics of many DL studies. We will discuss some of the PIECES items in the following sections.

PIECES recommends prospective registration of the study protocol similar to that of the RCT, using reduced publication bias as their main argument. Publication bias can limit progression in a field, since negative results, e.g. a certain network architecture for semantic segmentation performed poorly, are not reported, resulting in other research groups may try the same architecture later, unknowing of the fact that previous results are poor. Thus, preregistration might yield more rapid identification of promising systems, and thus increase progression in the field [8]. Another benefit of prospective registration, is that it facilitates thorough study design and planning. This can have very positive ef-

[6] **Interventional studies** are studies that involves the use of medication or procedures.

[7] **Publication bias** is the situation in which studies with "negative" results (wrt. to their *a priori* hypothesis) remain unpublished [53].

fects on a research project, in that potential issues and pitfalls which may arise can be identified ahead, and thus adjusted for. This thesis originally aimed at taking an exploratory approach to wrinkle segmentation on hyperspectral images. Thus, the results could have been used to design larger and more thorough studies, aiming at revealing true utility, in which the design could have been aided by prospective registration.

PIECES recommends external validation. The origin of the external cohort, an explanation to why it is regarded as external to the development cohort (see section 2.5.3), as well as precise criteria for inclusion and exclusion of participants/samples, together with a clear statement of the medical setting and target population that the external cohort represents, should be included in reports. In section 4.2, we primarily discussed internal validation within the development cohort, and how to optimize data utilization, wrt. both optimal performance and feasibility of error calculation. However, it is interesting to draw some lines between our efforts concerning internal validation, and that of external validation. External validation offers protection from data leakage, and it also offers a test on the robustness of the model; there might be certain features in the development cohort, for example systematic biases induced by acquisition equipment, that we cannot test for by internal validation. In case of our dataset, external validation could unravel biases introduced by the narrow development cohort, featuring mainly young, healthy and light skinned individuals with similar macro-structures. Moreover, with external validation it is easier to identify problems related to models utilizing unintentional and possibly false features [8]. Narla et al. [55] found that their model was more prone to classify skin lesions as malignant if there was a ruler besides the lesion. The reason was that that in their development cohort, lesions with rulers were in fact more likely to be malignant. The ruler is an example of a false feature – it obviously has nothing to do with whether the lesion is malignant or benign. As well as being a false feature, it is an example of a systematic bias that can occur within a cohort.

The implications of such false features and systematic biases, can be significant. For example, if at Hospital A, it is common practice to place rulers besides lesions if the clinician suspects cancer, and vice versa, but at Hospital B no rulers are used, the end model will probably under-estimate malignant lesions at Hospital B. At Hospital A on the other hand, the model will be guided by the clinician (in both directions). In the most tragic event, this false feature may be fatal. Thus, the importance of external validation should be evident.

PIECES further recommends to specify the acquisition of input data. This includes noting the expertise of any humans involved in the process, for example that a pathologist annotated the regions of interests in slide images [8]. In section 4.2 we discussed confidence in the ground truth, factors that impacted

human annotators, and subsequently the impacts of annotations on final model performance. Thus, additionally to what is already specified in the PIECES protocol, we recommend to clearly describe the annotation process, and preferably also include analysis of inter- and intra-reproducibility, as well as qualitative analysis on the overall quality of manual annotations.

Chapter 5

Conclusions and Final Remarks

5.1 Conclusions and Guidelines

The development of the Frangi-Gabor model was cumbersome, and it had the lowest performance in this thesis. Compared to various segmentation models, where one only needs to annotate masks and then train the model, *we conclude that it is indeed tempting to move from a classical approach to a ML approach*.

In the U-Net models, the dataset was strongly augmented to extend the number of training samples and introduce new variance. In Model A, the data was augmented and then divided into training, tuning and test sets. In Model B, featuring cross-validation, data augmentation was performed posterior to fold division. This uncovered huge discrepancy in the performance estimates between the two models, *confirming severe overfitting, data leakage and poor data augmentation in Model A*. Furthermore, it was found that the performance estimates of each fold of Model B was subject to great variability, suggesting that significant differences in the data distribution across training- and validation folds are present. *Hence, we conclude that the dataset is too small*, even for cross-validation. The insight gained by cross-validation into properties of the dataset wrt. data distribution can be important. We present the following guidelines:

Guideline 1 Divide into train, tuning and test set *prior* to data augmentation to avoid data leakage.

Guideline 2 If practical, use cross-validation.

Additionally to the use of cross-validation, and in accordance with the PIECES recommendations, external validation should be used.

Guideline 3 Use external validation.

The quality of manual annotations are important to the final model performance. The model can never claim better performance than the confidence one has in what is presented as ground truth to the algorithm. The confidence is typically calculated based on the inter- and intra-reproducibility of annotations. Furthermore, the annotation process is important to the quality of annotations. Poor annotations propagate as noise into the DL model and worsens performance. The PIECES protocol recommends to note the expertise of any human involved in the data acquisition process, for example that a pathologist annotated the regions of interest in slide images. We agree to the PIECES recommendation, but additionally, we recommend the following guidelines (guidelines 4–6):

Guideline 4 Always use multiple annotators in order to calculate inter- and intra-reproducibility.

Guideline 5 The annotation process must be designed carefully, and the annotation tools chosen likewise. The annotator must be sufficiently instructed into how annotations should be constructed. Make sure that the tools work as intended and do not put unnecessary restrictions on the annotator; minor niggles add to cumulative fatigue, and may propagate into the model resulting in decreased performance.

Guideline 6 Clearly describe the annotation process, in terms of equipment, software, expertise, whether or not the annotators could interact with experimenters during the annotation session, whether the room was quiet or not, whether or not the annotations were performed under time pressure or not, etc. Not clearly describing the annotation process can give irreproducible results.

5.2 Final Remarks

In this thesis, we walked through the full development process of a classical computer vision method, which in the end performed worse than the easier to use ML models. However, the ML model development was not a walk in the park: scarce data forced us to take on the methods of cross-validation to increase available training data, while still retaining the possibility of calculating unbiased estimates of generalization error. Cross-validation further showed that the impact of data leakage and poor data augmentation were significant. Moreover, the great variability across validation folds, suggests that the dataset was too small even for cross-validation.

Furthermore, we saw and discussed issues related to manual annotation, and

how one must take care in designing the annotation process, and we also discussed the importance of being transparent when reporting the process in order to avoid irreproducible results.

Guidelines coupled to each of the objectives proposed in section 1.2 were presented to assist developers in achieving increased validity and uncover validity in their machine learning studies.

The discussion, illustrations and final guidelines presented thus concludes this thesis, thereby realizing all the objectives, as well as the overall goal.

Bibliography

[1] Ng C, Yap MH, Costen N, Li B. Wrinkle Detection Using Hessian Line Tracking. IEEE Access. 2015;3:1079–1088.

[2] Keshet, Renato. User:Renatokeshet;. (accessed: 12.06.2021). Available from: https://en.wikipedia.org/wiki/User:Renatokeshet.

[3] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation; 2015.

[4] Grimstad B. TTK28, Lecture notes, '5. Hyper-parameters and validation sets'; 2020.

[5] Brunton SL, Kutz JN. Data-Driven Science and Engineering: Machine Learning, Dynamical Systems and Control. Cambridge University Press; 2019.

[6] ImageJ. Image Processing and Analysis in Java;. (accessed: 31.05.2021). Available from: https://imagej.nih.gov/ij/.

[7] iBiology com. Bioimage Analysis;. (accessed: 31.05.2021). Available from: https://www.ibiology.org/techniques/bioimage-analysis/.

[8] Kleppe A, Skrede OJ, De Raedt S, Liestøl K, Kerr DJ, Danielsen HE. Designing deep learning studies in cancer diagnostics [Journal Article]. Nature Reviews Cancer. 2021;21(3):199–211. Available from: https://doi.org/10.1038/s41568-020-00327-9.

[9] Nagendran M, Chen Y, Lovejoy CA, Gordon AC, Komorowski M, Harvey H, et al. Artificial intelligence versus clinicians: systematic review of design, reporting standards, and claims of deep learning studies. BMJ. 2020;368. Available from: https://www.bmj.com/content/368/bmj.m689.

[10] Singh, Ranjeet. Recent advances in modern computer vision;. (accessed: 06.06.2021). Available from: https://towardsdatascience.com/recent-advances-in-modern-computer-vision-56801edab980.

[11] Montagna, William and Ebling, F John. Human Skin in Encyclopedia Britannica;. (accessed: 01.06.2021). Available from: https://www.britannica.com/science/human-skin.

[12] Kollias N, Baqer AH. Absorption Mechanisms of Human Melanin in the Visible, 400–720nm. Journal of Investigative Dermatology. 1987;89(4):384–388. Available from: https://www.sciencedirect.com/science/article/pii/S0022202X87909675.

[13] Furue M, Mitoma C, Mitoma H, Tsuji G, Chiba T, Nakahara T, et al. Pathogenesis of systemic sclerosis - current concept and emerging treatments. Immunologic Research. 2017;65(4):790–797. Available from: https://doi.org/10.1007/s12026-017-8926-y.

[14] Ferreli C, Gasparini G, Parodi A, Cozzani E, Rongioletti F, Atzori L. Cutaneous Manifestations of Scleroderma and Scleroderma-Like Disorders: a Comprehensive Review. Clinical Reviews in Allergy & Immunology. 2017;53:306–336. Available from: https://doi.org/10.1007/s12016-017-8625-4.

[15] Frangi AF. Three-Dimensional Model-Based Analysis of Vascular and Cardiac Images;. Available from: https://dspace.library.uu.nl/handle/1874/377.

[16] Cula GO, Bargo PR, Nkengne A, Kollias N. Assessing facial wrinkles: automatic detection and quantification. Skin Research and Technology. 2013;19(1):e243–e251. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1600-0846.2012.00635.x.

[17] Ng C, Yap MH, Costen N, Li B. Automatic Wrinkle Detection Using Hybrid Hessian Filter. Cremers D, Reid I, Saito H, Yang MH (eds) Computer Vision – ACCV 2014 ACCV 2014 Lecture Notes in Computer Science Springer, Cham. 2015;9005.

[18] Elbashir RM, Yap MH. Evaluation of Automatic Facial Wrinkle Detection Algorithm. Journal of Imaging;6(4). Available from: https://doi.org/10.3390/jimaging6040017.

[19] Decenciére E, Belhedi A, Koudoro S, Flament F, François G, Rubert V, et al. A 2.5D Approach to Skin Wrinkles Segmentation. Image Analysis & Stereology;38(1). Available from: https://www.ias-iss.org/ojs/IAS/article/view/1925/1096.

[20] Frangi AF, Niessen WJ, Vincken KL, Viergever MA. Multiscale vessel enhancement filtering. In: Wells WM, Colchester A, Delp S, editors. Medical Image Computing and Computer-Assisted Intervention — MICCAI'98. Berlin, Heidelberg: Springer Berlin Heidelberg; 1998. p. 130–137.

[21] SciKit-Learn. skimage.filters.gabor;. (accessed: 08.12.2020). Available from: https://scikit-image.org/docs/dev/api/skimage.filters.html#skimage.filters.gabor.

[22] OpenCV. Histogram Equalization;. (accessed: 11.12.2020). Available from: https://docs.opencv.org/master/d5/daf/tutorial_py_histogram_equalization.html.

[23] OpenCV. OpenCV: Morphological Transformations;. (accessed: 03.06.2021). Available from: https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html.

[24] Buslaev A, Parinov A, Khvedchenya E, Iglovikov V, Kalinin A. Albumentations: fast and flexible image augmentations; 2018.

[25] Simard PY, Steinkraus D, Platt JC. Best practices for convolutional neural networks applied to visual document analysis. In: Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.; 2003. p. 958–963.

[26] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998;86(11):2278–2324.

[27] Albumentations ai. Albumentations: fast and flexible image augmentations;. (accessed: 06.04.2021). Available from: http://albumentations.ai/.

[28] Mansurov, Nasim. What is Lens Distortion?;. (accessed: 30.05.2021). Available from: https://photographylife.com/what-is-distortion.

[29] MathWorks. dice;. (accessed: 09.12.2020). Available from: https://se.mathworks.com/help/images/ref/dice.html.

[30] Eramian M, Power C, Rau S, Khandelwal P. Benchmarking Human Performance in Semi-Automated Image Segmentation. Interacting with Computers. 2020 08;32(3):233–245. Available from: https://doi.org/10.1093/iwcomp/iwaa017.

[31] MathWorks. jaccard;. (accessed: 09.12.2020). Available from: https://se.mathworks.com/help/images/ref/jaccard.html.

[32] Nielsen M. Neural Networks and Deep Learning. Determination Press; 2015. http://neuralnetworksanddeeplearning.com/.

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

[34] Vaddireddy H, Rasheed A, Staples AE, San O. Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensor observation data. Physics of Fluids. 2020;32(1):015113. Available from: https://doi.org/10.1063/1.5136351.

[35] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics. 2019;378:686 – 707. Available from: http://www.sciencedirect.com/science/article/pii/S0021999118307125.

[36] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. J Mach Learn Res. 2014 Jan;15(1):1929–1958.

[37] Microsoft Research/Kaggle. Dogs vs. Cats;. (accessed: 19.05.2021). Available from: https://www.kaggle.com/c/dogs-vs-cats.

[38] Stone M. Cross-Validatory Choice and Assessment of Statistical Predictions. Journal of the Royal Statistical Society: Series B (Methodological). 1974;36(2):111–133. Available from: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1974.tb00994.x.

[39] Breiman L. Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). Statistical Science. 2001;16(3):199 – 231. Available from: https://doi.org/10.1214/ss/1009213726.

[40] Martens HA, Dardenne P. Validation and verification of regression in small data sets. Chemometrics and Intelligent Laboratory Systems. 1998;44(1):99–121. Available from: https://www.sciencedirect.com/science/article/pii/S0169743998001671.

[41] Browne MW. Cross-Validation Methods. Journal of Mathematical Psychology. 2000;44(1):108–132. Available from: https://www.sciencedirect.com/science/article/pii/S0022249699912798.

[42] Kumar A. Importance of Being Unique From Finger Dorsal Patterns: Exploring Minor Finger Knuckle Patterns in Verifying Human Identities. IEEE Transactions on Information Forensics and Security. 2014;9(8):1288–1298.

[43] Pixelmator. Pixelmator for iOS;. (accessed: 15.04.2021). Available from: https://www.pixelmator.com/ios/.

[44] Milesial. Pytorch-UNet;. (accessed: 13.12.2020). Available from: https://github.com/milesial/Pytorch-UNet.

[45] Arlot S, Celisse A. A survey of cross-validation procedures for model selection. Statistics Surveys. 2010;4(none):40 – 79. Available from: https://doi.org/10.1214/09-SS054.

[46] Mosteller F, Tukey JW. Data analysis, including statistics. Handbook of social psychology. 1968;2:80–203.

[47] Bearman A, Russakovsky O, Ferrari V, Fei-Fei L. What's the Point: Semantic Segmentation with Point Supervision. In: Leibe B, Matas J, Sebe N, Welling M, editors. Computer Vision – ECCV 2016. Cham: Springer International Publishing; 2016. p. 549–565.

[48] Savran A, Sankur B, Taha Bilge M. Comparative evaluation of 3D vs. 2D modality for automatic detection of facial action units. Pattern Recognition. 2012;45(2):767–782. Available from: https://www.sciencedirect.com/science/article/pii/S0031320311003104.

[49] Bartle, Elinor. Importance of Randomized Control Trials;. (accessed: 08.06.2021). Available from: https://www.uib.no/en/cih/114638/importance-randomized-control-trials.

[50] BiopharmaInstitute. What Is The Difference Between Single Blind And Double Blind Clinical Trials?;. (accessed: 08.06.2021). Available from: https://www.biopharmainstitute.com/faq/what-is-the-difference-between-single-blind-and-double-blind-clinical-trials.

[51] JMIR. Does my trial (RCT) have to be registered?;. (accessed: 08.06.2021). Available from: https://support.jmir.org/hc/en-us/articles/115001389307-Does-my-trial-RCT-have-to-be-registered-.

[52] U S NIH. How to register your study;. (accessed: 08.06.2021). Available from: https://clinicaltrials.gov/ct2/manage-recs/how-register.

[53] Sun J, Freeman BD, Natanson C. Chapter 22 - Meta-analysis of Clinical Trials. In: Gallin JI, Ognibene FP, Johnson LL, editors. Principles and Practice of Clinical Research (Fourth Edition). fourth edition ed. Boston: Academic Press; 2018. p. 317–327. Available from: https://www.sciencedirect.com/science/article/pii/B9780128499054000228.

[54] Harvey H, Oakden-Rayner L. Guidance for Interventional Trials Involving Artificial Intelligence. Radiology: Artificial Intelligence. 2020;2(6):e200228. Available from: https://doi.org/10.1148/ryai.2020200228.

[55] Narla A, Kuprel B, Sarin K, Novoa R, Ko J. Automated Classification of Skin Lesions: From Pixels to Practice. Journal of Investigative Dermatology. 2018;138(10):2108–2110. Available from: https://www.sciencedirect.com/science/article/pii/S0022202X18322930.

[56] Randeberg LL. Hyperspectral characterization of tissue in the SWIR spectral range: a road to new insight? . In: Alfano RR, Demos SG, Seddon AB, editors. Optical Biopsy XVII: Toward Real-Time Spectroscopic Imaging and Diagnosis. vol. 10873. International Society for Optics and Photonics. SPIE; 2019. p. 125 – 140. Available from: https://doi.org/10.1117/12.2504297.

[57] Paluchowski LA, Milanic M, Bjorgan A, Grandaunet B, Dhainaut A, D MHM, et al. Identification of inflammation sites in arthritic joints using hyperspectral imaging. In: Farkas DL, Nicolau DV, Leif RC, editors. Imaging, Manipulation, and Analysis of Biomolecules, Cells, and Tissues XII. vol. 8947. International Society for Optics and Photonics. SPIE; 2014. p. 60 – 66. Available from: https://doi.org/10.1117/12.2040499.

[58] Vasefi F, MacKinnon N, Farkas DL. Chapter 16 - Hyperspectral and Multispectral Imaging in Dermatology. In: Hamblin MR, Avci P, Gupta GK, editors. Imaging in Dermatology. Boston: Academic Press; 2016. p. 187 – 201. Available from: http://www.sciencedirect.com/science/article/pii/B9780128028384000169.

[59] Austnes B. kidneb7/hyperspectral-image-cropper;. A simple Python tool based on OpenCV and Spectral Python for cropping datacubes of hyperspectral images. Available from: https://github.com/kidneb7/hyperspectral-image-cropper.

Appendix A

Hyperspectral Imaging

Hyperspectral Cameras

Hyperspectral imaging (HSI) is an imaging technique where we are sampling wavelengths at much denser intervals compared to that of ordinary imaging, where we only sample three wavelengths. Thus, with HSI we obtain the full spectrum of the scene within the range in question. Hyperspectral cameras are often specialized equipment specifically ordered for the problem they are intended to solve, hence hyperspectral cameras exist with a wide range of spectral bands, everything from 10s – 1000s of spectral bands in various wavelength segments.

In medicine, HSI has for example been used for characterization of tissue in the short wave infrared (SWIR) range (950-2500 nm) [56] and identification of inflammation sites in arthritic joints in the visual and near-infrared (VNIR) and SWIR range (400-1700 nm) [57].

For separating wavelengths, optical components such as prisms and diffraction grating are used [58]. When light is passed through these components, the various wavelengths are bended differently. This is illustrated in fig. A.0.1. The separated wavelengths are then pointed at a photosensitive chip, and the intensity of each wavelength is determined and recorded.

Some hyperspectral cameras are line scan cameras [58]. Hence, instead of having a 2D image sensor, they scan one line in the scene and then physically move a short distance for the next line scan. Capturing an image can therefore be a time consuming process.

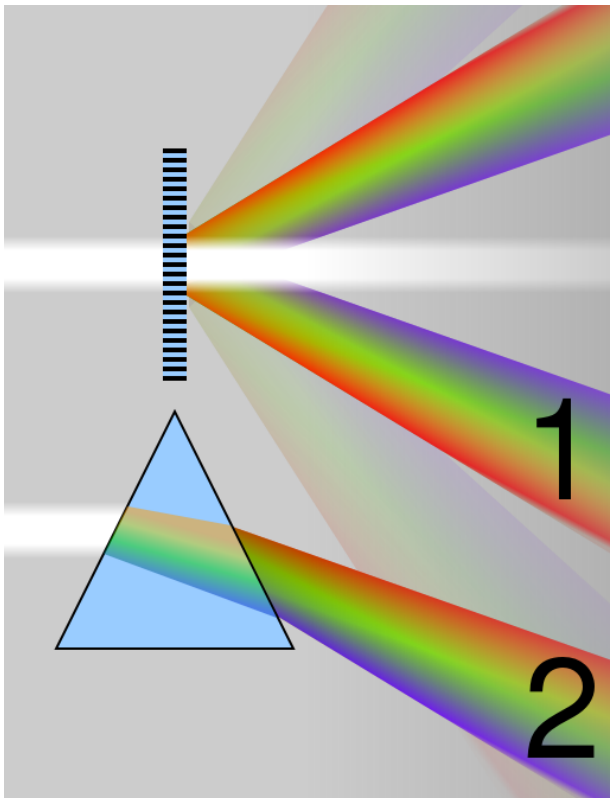

Figure A.0.1: Wavelength separation by diffraction grating (1) and prism (2). Original figure by Cmglee, published under licence CC BY-SA 3.0, via Wikimedia Commons. Figure has been slightly modified.

HSI Dataset

Hyperspectral images of the hands of 29 healthy individuals were collected by St. Olavs Hospital. The middle finger PIP joint area from each individual was extracted using our own custom tool, which is available from GitHub [59]. It is important that all images in the dataset maintain the same dimensions. By scanning through all cropped segments, we found the minimum widths and heights in the dataset. Following, each cropped sample were cropped again to match a specific aspect ratio of 0.89 (W/H) within the maximum limits of widths and heights. The specific aspect ratio is the same aspect ratio as of that from the Kumar dataset. After processing, each middle finger PIP sample had the dimensions 77×86 px. A preview of the hyperspectral dataset is shown in fig. A.0.2.

Using cropping for achieving homogeneous dimensions was chosen to avoid distorting spatial and/or spectral features in the original images.

We chose not to include the PIP segments of the index finger, ring finger or little

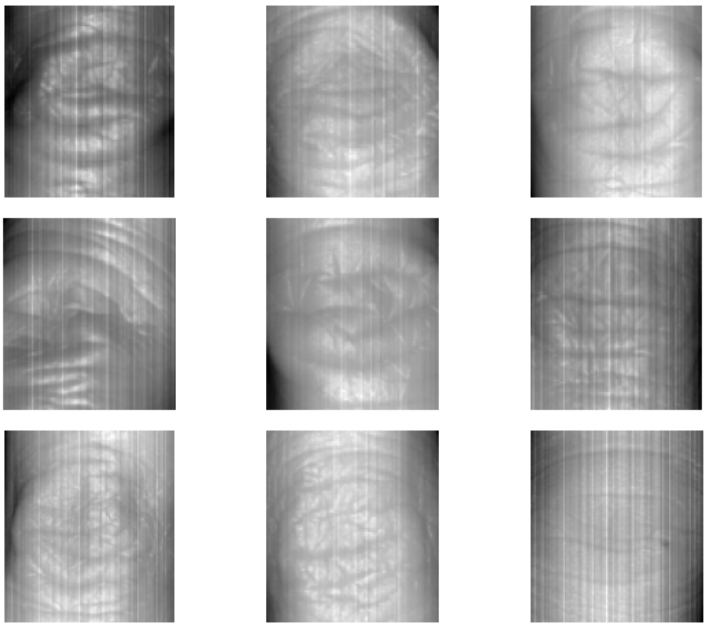

Figure A.0.2: A preview of the hyperspectral dataset.

finger, since this would force the dimensions of the cropped dataset to be even smaller.

The software Pixelmator 2.6.3 [43] was used for manual annotation on an Apple iPad Air 4. A single image was loaded into the software. Then, a new layer was added. Using the `Paint/Pixel` tool, wrinkles were traced to the best of the author's abilities. When all the wrinkles were traced, the `Fill` tool was used to overlay each trace with white color. Then, a new black layer was added, and the white traces were moved to the front. An Apple Pencil 2 was used for drawing. Figure A.0.3 shows the manual annotation in progress.

A preview of the HSI dataset is given in fig. A.0.2 where the images have been converted to RGB by band 180.

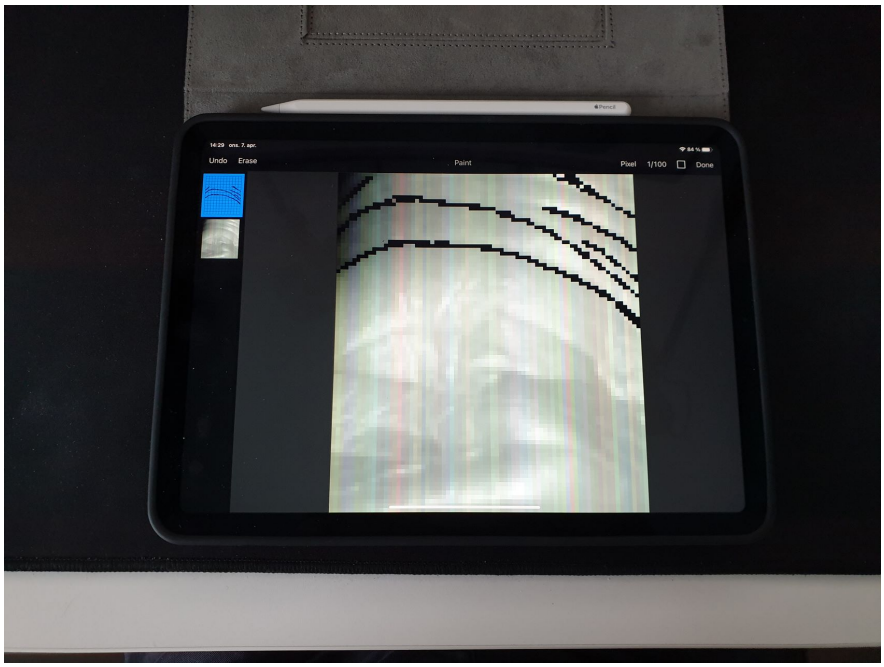

Figure A.0.3: Manual annotation of HSI dataset in Pixelmator.

Appendix B

Extra Material

This appendix lists extra material supporting the material in the thesis.

Frangi-Gabor Extended Results

Frangi-Gabor performance on 6 random samples from the Kumar test set are shown in fig. B.0.1.

U-Net Model on Manually Annotated HSI

The U-Net was modified to take hyperspectral images as inputs. `RMSprop` optimizer, `batch size=1`, `LR=0.0001`, `epochs=100`.

The best 3 predictions with a U-Net model on the HSI test set are shown in fig. B.0.2.

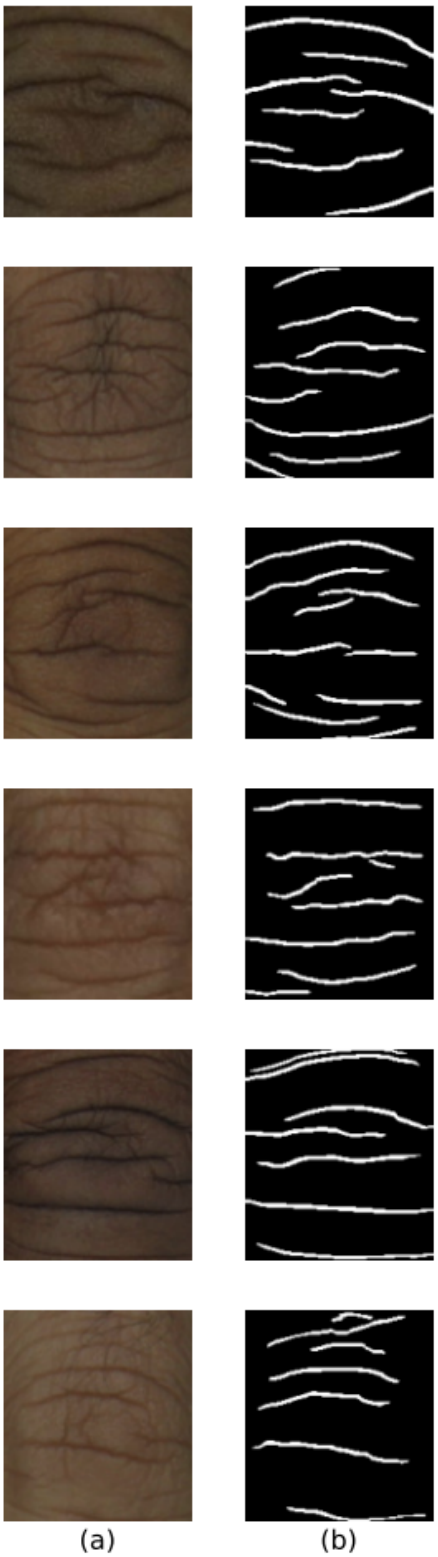

Figure B.0.1: Predicted wrinkles for six random samples from the Kumar test set. (a) Original input images are shown in the left column. (b) Predictions from the Frangi-Gabor algorithm are shown in the right column.

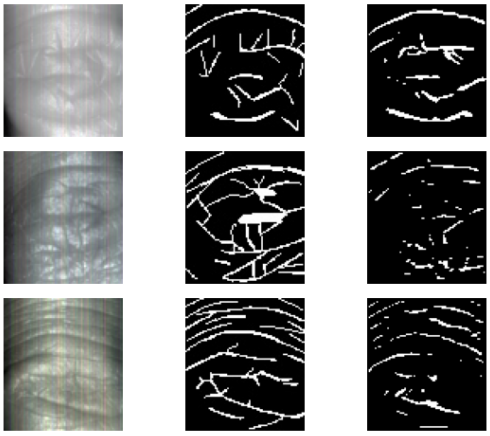

Figure B.0.2: U-Net model for HSI. Left column shows pseudo-color HSI input images, middle column shows the manual masks, and the right column shows predictions from the model.

Bendik Austnes

Increasing Validity and Uncovering Utility in Machine Learning Studies

NTNU
Norwegian University of
Science and Technology