TUHH
Hamburg
University of
Technology

Lennard Korte

Impact of different Data Augmentation Techniques for Deep Learning with Optical Coherence Tomography

Institute of Medical Technology
and Intelligent Systems
Building E | 21073 Hamburg
www.tuhh.de/mtec

Institute of Medical Technology and Intelligent Systems

A bachelor thesis written at the Institute of Medical Technology
and Intelligent Systems and submitted in partial fulfillment of the requirements for the degree Bachelor of Science.

Author: Lennard Korte

Matriculationnumber: 53673

Study Program: Computer Science

Title: Impact of different Data Augmentation Techniques for Deep Learning with Optical Coherence Tomography

Date: May 30, 2022

Supervisors:	Robin Mieling (MSc.)
	Alexander Schlaefer (Dr.-Ing.)
Referees:	Prof. Dr.-Ing. Alexander Schlaefer
	Till Robin Mieling (B.Sc)

Contents

List of Figures

List of Tables

Acronyms

1D one-dimensional. 16

2D two-dimensional. 4, 16, 25, 31, 45

3D three-dimensional. 4

ACC accuracy. 17, 18, 21, 29, 41, 43, 44, 47, 55

BACC balanced accuracy. xiii, 18, 19, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56, 58, 60, 62, 63

CLAHE contrast limited adaptive histogram Equalization. vii, 39, 40, 42, 51, 52

CNN convolutional neural network. 1, 2, 14, 15, 16, 22, 27, 28, 29, 34, 38, 40, 43, 60

CV cross-validation. vii, 21, 22, 40, 46, 47, 56, 57, 63

DA data augmentation. vii, xiii, 2, 3, 20, 22, 23, 24, 27, 28, 29, 31, 34, 35, 37, 40, 41, 46, 54, 55, 56, 57, 59, 62, 63

DL deep learning. xiii, 1, 2, 3, 9, 19, 20, 22, 25, 27, 28, 63

DOR diagnostic odds ratio. 19

FD-OCT frequency domain optical coherence tomographie. 4, 6, 7

FDR false discovery rate. 18

FOR false omission rate. 18

GAN generative adversarial network. 12, 61

IVOCT intravascular optical coherence tomographie. vii, xiii, 1, 2, 3, 6, 7, 23, 24, 26, 27, 30, 31, 32, 33, 34, 35, 37, 39, 42, 43, 45, 47, 49, 50, 55, 56, 57, 58, 59, 60, 61, 63

IVUS intravascular ultrasound. 1

MCC Matthews correlation coefficient. xiii, 19, 44, 45, 47, 48, 49, 50, 51, 52, 53, 54, 56, 57, 58, 60, 61, 62, 63

ML machine learning. 7, 8, 9, 10, 12, 14, 16, 19, 20

MSE mean squared error. 13

NN neuronal network. 8, 9, 10, 11, 12, 14, 15, 25, 27, 37, 63

NPV	negative predictive values. 18
OCT	optical coherence tomographie. vii, 1, 3, 4, 5, 6, 7, 23, 25, 31, 61
PPV	positive predictive values. 18, 19, 45, 47, 48, 49, 50, 51, 52, 53, 54
ReL	rectified linear activation function. 11
ReLU	rectified linear activation function unit. 11, 12
ResNet	residual neural network. vii, 1, 3, 15, 16, 28, 29, 32, 46, 47, 48, 49, 52, 61, 63
ROC	receiver operating characteristic. 18
SD-OCT	spectral domain optical coherence tomographie. 4
SENS	sensitivity. 17, 18, 19, 45, 47, 48, 49, 50, 51, 52, 53, 54, 55, 60
SPEC	specificity. 17, 18, 45, 47, 48, 49, 50, 51, 52, 53, 54, 55, 60
SS-OCT	swept source optical coherence tomographie. 4
TD-OCT	time domain optical coherence tomographie. 4, 6, 7

Glossary

Affine transformation Is a geometric transformation that does not necessarily preserve distances and angles, but lines and parallelism. 23, 24

A-scan Is a one-dimensional Depth profile. 4, 6, 7, 23, 31

Atherosclerosis Is the thickening or hardening of the arteries, that is caused by a buildup of plaque in the inner of an artery. xiii

Batch normalization Is a method of making artificial neural networks faster and more stable by normalizing layer inputs through rescaling and rescaling. 15, 16

B-scan Is a two-dimensional image, sequence of A-scans shifted along a lateral spatial dimension. vii, 4, 30, 31, 32, 33, 37, 38, 41, 42, 47, 60, 63

C-scan Is a three-dimensional image, sequence of B-scans shifted along third spatial dimension. 4

Debian Is a free and open-source Linux distribution, usually known as Debian GNU/Linux, that was created by the community-supported Debian project, founded by Ian Murdock. 27

Generalizability Generalization refers to your model's ability to adapt properly to new, previously unseen data, drawn from the same distribution as the one used to create the model [41]. 20

Heat map Is a representation of data in the form of a map or diagram in which data values are represented as colors [71]. 16

Histogram Is a graphical representation of the images intensity distribution. It visualizes the number of pixels for each intensity value. vii, 38, 39, 40

ImageNet Is a large visual database designed by a number of academic contributors for use in visual object recognition software research. 32, 33, 41, 43, 45

Interpolation Is a method of finding new data points based on the range of a discrete set of known data points. To generate a simplified function that is still quite close to the original a data points from the original function are interpolated. 6, 23, 32, 41

Learning rate Is a hyper parameter that controls how much weights and biases of a neural network are adjusted during training with respect to the error gradient. vii, 14, 29, 40, 41, 43, 44, 57

Logistic regression Is a technique for modeling the probability of a discrete outcome as a function of an input variable. The most common logistic regression models a binary outcome with help of the Sigmoid function in contrast to linear regression often used for continuous variable estimation [30]. 12

Makima method Is a MATLAB-specific modification of Akima's derivative formula and shortcut for modified Akima piecewise cubic Hermite interpolation. 31

M-scan Is a two-dimensional image, but temporal sequence of A-scans. 4

Mutual information Is a measure of image matching, that expresses the amount of information given by image Y on image X. A value of 1.0 would reflect perfect agreement and 0.0 complete disagreement between the two images X and Y.. 59

Optimizer Is an algorithm that modifies the weights, biases and learning rate of a neural network to minimize the cost function (or error) and thus improves prediction performance. Thus, its choice is a key aspect in having a successful training of a NN. 14

Overfitting Is a modeling error in statistics that occurs when a function is too closely aligned to a limited set of data points. As a result, the model is useful in reference only to its initial data set, and not to any other data sets. [97] In Machine Learning it refers to high error with respect to test data as well as low prediction error with respect to training data. 2, 20, 21, 22, 26, 56, 59

Regression Is a statistical method that attempts to assess the strength and character of the relationship between a dependent variable and a set of other variables [7]. 7, 8, 10, 11, 13, 16, 21

WandB Is a central dashboard that allows you to keep track of your hyperparameters, system metrics, and forecasts in real time, allowing you to observe and compare models in real time. 30

Weighted random sampler Is a sampling algorithm, that is able to select a sample from a given data set subdivided into k classes. The specified selection probabilities can be unequally distributed such that one can counteracting imbalances for further processing. 31

Abstract

Atherosclerosis is one of the leading but preventable and treatable causes of death worldwide. For effective treatment, e.g. choice of stent size and its positioning, it is essential for the attending physician to have an accurate idea of the intravascular anatomy. Lesions (plaque) and arterial walls can be visualized with intravascular optical coherence tomographie (IVOCT), a high-resolution imaging technique, that can be applied by experts. Due to the high number of images captured during clinical routines, research spends tremendous efforts on automatic plaque detection (image classification) with deep learning (DL) in polar as well as cartesian representation for fast and accurate decision support. Data augmentation (DA) is an essential part of training DL models in machine learning. A variety of augmentation strategies, including cropping, flipping, and convolutional filters, have been proposed. They have shown to maintain important characteristics of IVOCT images without handcrafting features, especially when in cartesian representation. While data augmentation has been commonly used for deep learning in IVOCT imaging, little work has been done to determine which augmentation techniques capture medical image statistics best and lead to a higher model performance. This work provides an overview about conventional techniques and proposes new, advanced techniques tested on a data set consisting of labeled in vivo patient images. Furthermore, augmentation techniques are compared and those that add value are proposed for further investigation building domain specific training strategies. Finally, it was shown that the models performance is determined by the extent to which relevant information and features of the original medical images are preserved in the augmented training sets. Overall, we find that a model trained with data that are augmented by the discussed techniques, for instance shifting or shearing, may boost the performance by 0.05 in Matthews correlation coefficient (MCC) and 0.03 in balanced accuracy (BACC). On the other hand, less effective transformations such as contrast jittering may deteriorate scores significantly by more than ten percent. Our results show that the right DA strategy is essential for performance optimization when building deep learning-based clinical decision support systems for plaque detection.

1 Introduction

Coronary heart disease is one of the leading causes of death [78, 47] accounting for more that 15% of all deaths in 2015 [57], despite being preventable and treatable. To treat the causes of coronary heart disease, optical coherence tomographie (OCT) technology can be used to allow a detailed examination of the in vivo tissue anatomy and coronary procurement from within.

In comparison to the typically used intravascular ultrasound (IVUS) technology, intravascular optical coherence tomographie (IVOCT) offers a higher micrometer resolution utilizing interferometry low-coherent light [81, 27]. Both imaging techniques that can be used for planning stent interventions or during interventions to guide stent placement and characterize atherosclerotic plaque types [57]. IVOCT is well suited for embedding in thin medical instruments such as catheters, because the infrared light can be transmitted via glass fibres [48, 36]. Due to its low penetration depth capabilities, it only allows visualization of regions directly around the catheter. This is sufficient for the attending physicians to obtain a detailed understanding of the region of interest, i.e. the arterial obstruction. [36] Currently IVOCT is the only imaging technique that enables identification of the most rupture susceptible thin-cap fibroatheroma [43].

A key problem is the large number of frames (often > 500) arising from single 2s to 5s pullbacks in IVOCT interventions [57], that cannot be reviewed by the practitioner during clinical routine. Automatic plaque detection is of major importance for a fast and accurate decision support system [36]. These must offer reliable information on the presence and nature of stenosis [36]. Various approaches for automatic IVOCT data analysis have been proposed. Correlating the images to histology data proofs general feasibility of inferring plaque characteristics from IVOCT data [106].

Methods for automatic IVOCT data analysis like stent detection [109, 64] and automatic lumen segmentation [43, 96] have been proposed. Automatic classification approaches using texture and optical properties as features have been proposed for plaque detection and segmentation [98, 16]. DL approaches have been proposed more recently for IVOCT data processing [1], in particular convolutional neural networks (CNNs) have shown remarkable success in image analysis tasks such as segmentation [63] and classification [35]. Even if plaque detection with CNNs has been employed mostly in preliminary studies, they provide promising initial results. Recent standard architectures such as Inception or residual neural networks (ResNets) have shown significant performance improvement for medical image analysis [63, 35]. The approach using CNNs in medical diagnostic tasks offers the advantage that we can omit the complicated and error-prone step of tissue segmentation. Progress has been made in using DL for image classification, especially in the prediction and segmentation of cancerous masses in lung, liver or breast scans [21, 23, 32].

Nevertheless, various shortcomings still need to be addressed. Polar images are often converted to cartesian representation for easier and more intuitive interpretation by practitioners. One of the questions that remains unanswered is whether either or both images (e.g. in two path deep learning (DL) architectures) are the more advantageous choice for training the DL models [35] and relevant for future use. The lack of large labeled image data sets from IVOCT interventions is one of the major challenges in the process of creating DL models that achieve the desired and necessary performance. Nevertheless the effect of having only a limited number of scans can be mitigated by a number of methods to a certain extend, for instance normalization or dropout that prevent the model from overfitting. DA denotes one of the advanced techniques that allow DL models to extract features from small data sets more effectively.

Hence, the main objective of this bachelor's thesis was to implement a DL pipeline to investigate the effects of DA techniques tailored to polar as well as cartesian representation. It was examined which DA techniques have positive effects on the overall model performance, that is, the extent to which augmented data sets retain properties of the original medical images. By conducting a number of experiments we investigate which DA techniques are recommended to use. Their individual as well as combined performance impact on the CNN has been investigated to form a strategy applicable in the domain of IVOCT imaging. In addition, the question is answered to what extent data augmentation techniques applied on IVOCT data can improve the networks performance. New IVOCT specific DA techniques are presented and included in the analysis. It is assumed that IVOCT images do not vary in their fundamental structure, such that observations can be transferred to other IVOCT data sets. Trained experts labeled the frames of each pullback with binary class characterizations that distinguished between the presence and absence of plaque.

2 Theoretical Concepts

In the following fundamental concepts of the underlying experiments are described and explained in detail. Beginning with the general concepts of image acquisition with IVOCT, we continue by laying out how DL models work. It is explained how the behaviour and performance of utilized ResNets may be interpreted. In addition to that, an overview of frequently used DA techniques is provided in the context of DL with IVOCT data.

2.1 Optical Coherence Tomography

OCT is a medical imaging technique for translucent or opaque materials. It works by generating an interference pattern between light traveling through the sample arm and light propagating in the reference arm of an interferometer to measure the time-of-flight of light returned from tissue [111, 27].

In an early stage OCT images were applied to display retinal structures, but later it was also employed in diverse applications like ophthalmology, where it is used to obtain detailed images from within the retina, and optometry for painting analysis [111, 27]. Oncology and dermatology are other common areas where OCT is used primarily for cancer detection [104, 111]. In cardiology it helps to diagnose and treat coronary artery diseases [48]. Technologies like medical ultrasonography, confocal microscopy, magnetic resonance imaging and OCT are differently suited for morphological tissue imaging. They primarily differ in resolution and depth of view. OCT has achieved a high micrometer resolution (less than 10µm axially and 20µm laterally) [77, 92, 48, 58]. It allows a penetration depth of a few millimeters below the surface in biological tissue [111, 27]. This is because the proportion of escaping light without scattering becomes too low in deeper sample layers so that it can be extracted [9, 27]. Other key advantages are the ability to rapidly acquire live images directly without preparation or direct contact to the subject. Furthermore OCT is non invasive and non ionizing [46].

2.1.1 Principle

OCT employs near-infrared light with long wavelength generated by super-luminescent diodes, which allows it to penetrate into the scattering medium. The architecture illustrated in fig. 2.1 is based on low-coherence interferometry (Michelson interferometer) owing to the use of light with broad bandwidths [27, 48]. The optical setup typically consists of the components described as follows. Initially, light from the source is split into two beams, first the sample arm targeted at object of interest, and second, a reference arm with a mirror. The combination of light reflection from both arms generates an interference pattern. Depending on the sample properties, areas with greater light reflection will create greater interference and vice versa. Interferometry allows direct detection of reflections, considering that the speed of light is too fast to measure it directly. The low coherence characteristic of the source light will ensure that reflections

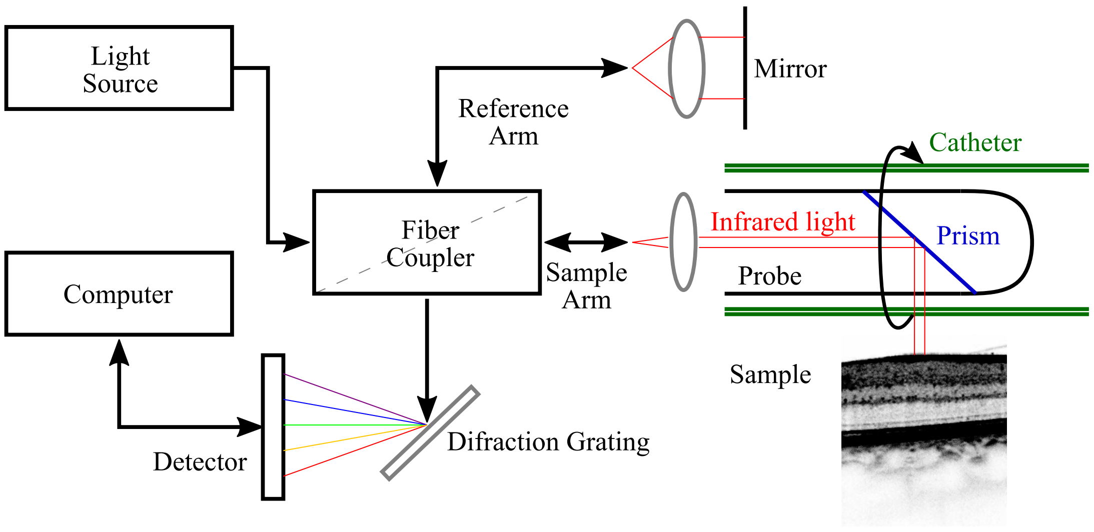


Fig. 2.1: Basic architecture of an FD-OCT system with a Michelson interferometer [27] at its core. The sample arm has been extended by a probe guided by the catheter and directs its beam on to a tissue [4].

outside the coherence length will not interfere [9]. Measuring reflection intensities enables the location of structures in spacial dimension of the axial depth scan (A-scan). When plotting an A-scan against the additional time dimension, we refer to it as M-scan. By adjusting the sample beam position we combine multiple A-scans to a series resulting in a cross-sectional tomogram referred to as B-scan (2D) or C-scan (3D), see fig. 3.2. [27, 9]

Time domain OCT (TD-OCT) image acquisition is highly limited in recording speed due to its scanning rate. By moving the mirror and thus changing its path length in the reference arm a reflective profile of the sample can be scanned [27]. The later introduced frequency domain OCT (FD-OCT) with a fixed mirror offers higher sampling rate and increased sensitivity [89]. FD-OCT resolves different axial positions based on the interference spectrum acquired with spectrally separated detectors. An increasing number of detection elements reduces losses and signal to noise ratio proportionally. While the scanning range is limited by parallel detection of multiple wavelength ranges, the spectral bandwidth defines the axial resolution. [39] We further distinguish between the two specialisations: spectral domain OCT (SD-OCT) and swept source OCT (SS-OCT). SD-OCT utilizes a broad light source with a spectrometer measuring all echo times simultaneously. Echo intensity distribution is resolved in space rather than in time [53]. An A-scan is made up by the accumulation of all axial measurements of the echo time delay. Special dispersive detectors are used to separate wavelengths. [9] In contrast SS-OCT encodes optical spectral frequency generated by a swept source laser. This enables the spectral measurement over time acquired by a spectral scanning source. Other known approaches are line field OCT using a broadband laser and line detection camera [29, 79] and full field OCT, where images are acquired orthogonal to the light beam of illumination [6].

Subsequently, we describe the axial ranging with low-coherence interferometry analogous to Drexler [27]. The following abstract will give an introductory overview of the most fundamental concepts in OCT technology. They have been described in papers by Billie and Drexler in detail [9, 27]. Assuming a known spectrum of the light source, with wavenumber k and an angular frequency of the light source, we can determine the samples reflective properties. The electrical field at time t and distance z with field amplitude $s(k,w)$ is defined as

$$E_i = s(k,w)e^{i(kz-wt)}. \tag{2.1}$$

The wavelength-independent beam splitter with a ratio of 0.5 ideally splits the source light into two equal parts. With a reflector distance to the beam splitter of z_R and a electric field reflectivity of r_R, the returned energy can be described as:

$$E_R = \frac{E_i}{\sqrt{2}} r_R e^{i2kz_R}. \tag{2.2}$$

The sample has a depth dependent and continuous electric field reflectivity profile $r_s(z_s)$. Note that z_s denotes the pathlength variable from the beam splitter along the beam axis. The reflected energy can be described with a convolution $(*)$ by:

$$E_S = \frac{E_i}{\sqrt{2}} \left[r_S(z_S) * e^{i2kz_S} \right] \tag{2.3}$$

For illustrative purposes we describe the sample reflection properties as a layered model r_S. Thus, we abstract it as series of N discrete real delta functions of the form

$$r_{\mathrm{S}}(z_{\mathrm{S}}) = \sum_{n=1}^{N} r_{\mathrm{S}n} \delta(z_{\mathrm{S}} - z_{\mathrm{S}n}) \tag{2.4}$$

Furthermore we have r_{Sn} as electric field reflectivity and z_{Sn} as pathlength from the beam splitter at layer n. After the returning fields passed through the beamsplitter a second time, their power is halved. They interfere at the square-law detector that generates a photocurrent according to its responsivity ρ:

$$I_D(k,w) = \frac{\rho}{2} \left\langle \left| \frac{s(k,w)}{\sqrt{2}} r_R e^{i(2kz_R - wt)} + \frac{s(k,w)}{\sqrt{2}} \sum_{n=1}^{N} r_{Sn} e^{i(2kz_{Sn} - wt)} \right|^2 \right\rangle. \tag{2.5}$$

Taking into account that a detector in praxis is not able to detect the temporal angular frequency w due to its response time, we can neglect the temporal invariant terms. With the help of the Eulers rule and by substituting the power spectral dependence of the light source $S(k) = \langle |s(k,w)|^2 \rangle$, the equation can then be rewritten as a sum of three components:

$$\begin{aligned} I_D(k) = & \frac{\rho}{4} \left[S(k) \left(R_R + \sum_{n=1}^{N} R_{Sn} \right) \right] \text{ DC Terms,} \\ & + \frac{\rho}{2} \left[S(k) \sum_{n=1}^{N} \sqrt{R_R R_{Sn}} \left(\cos[2k(z_R - z_{Sn}]) \right) \right] \text{ Cross-correlation Terms and} \\ & + \frac{\rho}{4} \left[S(k) \sum_{n \neq m = 1}^{N} \sqrt{R_{Sn} R_{Sm}} \left(\cos[2k(z_{Sn} - z_{Sm}]) \right) \right] \text{ Auto-correlation Terms} \end{aligned} \tag{2.6}$$

First, the DC term depends on power reflectivity of the mirror and is proportional to the source wavenumber spectrum. Sample reflectivity information are provided by the cross-correlation term. It depends on the pathlength difference between sample reflectors and reference arm, as well as the light source wavenumber and contains the desired reflection profile information. Additionally the often comparatively small autocorrelation term represents artifacts created by interference occuring between sample reflectors.

Note that the light source spectrum $S(k)$, characterized by the wavenumber bandwidth δk, is the inverse of the Gaussian-shaped coherence function $y(z)$ characterized by the coherence length l_c. The coherence length is a function of the light source bandwidth. [53]

$$l_c = \frac{2\sqrt{\ln(2)}}{\Delta k} = \frac{2\ln(2)}{\pi}\frac{\lambda_0^2}{\Delta\lambda} \tag{2.7}$$

2.1.2 Image Reconstruction

The raw detector signal can be reconstructed by first subtracting the known sensor specific offset and removing the DC term, which can be determined before actual measurement. In TD-OCT the wavenumber-dependent detector current is captured on single receiver. Thus we obtain the result by integrating at each reference beam length k_Z over all k. In comparison, FD-OCT detects the spectral signal $I_D(k)$ and acquires the depth information by applying the Fourier transformation. The inverse Fourier transform should map a signal from wavenumber k to physical distance z, but spectral decomposition is not necessarily linear in k-space. Therefore interpolate the wavenumber to new coordinates, also referred to as dechirping. The spectrum is linearized by mapping to corresponding non-linear and equidistant interpolation intervals. We multiply the signal with a Hann-window to proportionally reduce the lateral deflections minimizing spectral leakage (apodisation). For interpretation as image, the complex magnitude or depth of the interferometric signal is considered, that is often logarithmically compressed. By applying the inverse Fourier transformation to the source spectrum, we obtain the desired depth information. This so called OCT A-scan with grey-scaled intensity data represents the scanned structures. The resolution is given by the coherence length along the depth axis. Laterally, the shape of the light beam is determined by the optical setup. The complex valued signal can be represented by normalizing the complex valued signal to the amplitude

$$|i(z)| = \sqrt{\mathrm{Re}^2\{i(z)\} + \mathrm{Im}^2\{i(z)\}}. \tag{2.8}$$

2.1.3 IVOCT

Cardiovascular or intravascular OCT is a catheter based imaging technique of in vivo coronary arteries and deployed stents (see fig. 2.1). It serves as a special form of OCT and was established as a high definition alternative to intravascular ultrasound with a slightly shorter field of view [52, 92]. Semi-automated imaging analyses permits acquisition of a number of accurate measurements like stent apposition and neointimal thickening that can assist cardiologists while planning or executing interventions [48]. Since coronary heart disease is one of the most common treatable and preventable causes of death, there is a strong demand for reliable treatment methods. IVOCT can be used by experts to detect and characterize lesions and plaque deposits on the arterial wall. In practice hundreds of images are acquired in each recording raising the demand for automatic

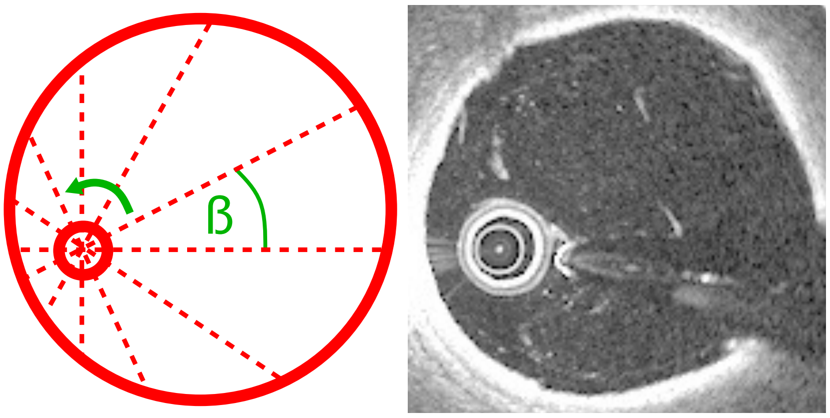


Fig. 2.2: Cross-sectional view showing a probe inside a catheter with a rotational pull-back as part of an IVOCT system. Between every A-scan, the probe rotates by an angle of β. Optimally smaller angles allow for a higher lateral resolution of the image (for illustrative purposes here transformed to the cartesian coordinate system).

decision support for plaque detection. IVOCT requires a single fiberoptic wire at the sample arm that emits light from the source and records its reflection. [48] The light is guided through the probe which itself is placed inside the catheter with 0.8 to 1.0 mm diameter [48]. At the tip of the probe is a prism positioned that redirects the beam focused with a given angle directly on the object of interest, i.e. an inner vessel wall. The axial resolution is determined by the light source wavelength of $1,250$ to $1,350nm$ [48]. Lateral resolution is determined by the spot size focused with the spot arm lens (see fig. 2.2). The wire simultaneously rotates while being pulled back along the artery at calibrated distances. We note that only the speed of light ($3 \times 10^8 m/s$) enables accurate measurements when concurrently moving the wire in space with a comparatively slow speed [48]. A frequency domain OCT system can reach pullback speeds of up to 20 millimeters per second and a rotation rate of up to 200 frames per second [48]. With a penetration depth of one to three millimeters, multiple axial A-scans can be continuously acquired by reflections recorded from the vessel wall (assuming a blood free environment) [35]. Recent versions of IVOCT make use of FD-OCT (instead of TD-OCT) with a fiberoptic coupler similar to a beam splitter. FD-OCT enables us to circumvent the limitations of generating a fast optical mirror delay. Thus we can reach a significantly higher imaging frequency with up to 100 frames per second. [48, 36]

2.2 Machine Learning

ML is a subset of artificial intelligence. It has been around since the 1950s, but only the low-cost as well as high-performance computing power and mass storage capabilities of recent years have made its use affordable. Along with this comes the increasing ease of collecting large amounts of data and the proliferation of data science. The broad field of ML, generally focuses is on predicting an outcome based on the available data. It automatically maps an output to an input. ML algorithms can save huge amounts of resources like time and money by solving specific tasks. There are other areas where automated information extraction and classification as well as regression can help to make processes more efficient. It can create new opportunities or substitute previous operations in an inestimable number of ways.

2.2.1 Universal Approximators

In contrast to rule-based decision making analysis ML algorithms excel at tasks for which functions are hard to define manually and patterns can not easily be spotted by human assessment, even professionals. As a form of universal approximators they can be utilized when patterns and the related decisions depend on a tremendously high number of input variables. According to universal approximation theory neuronal networks (NNs) can approximate any type of function as long as enough nodes or layers are available [5]. State-of-the-art NNs offer the possibility to approximate many functions accurately even with a limited number of nodes. This opened a completely new field of use cases for seemingly intelligent and automatic computer programs with practical implications of ML across multiple industry sectors, including healthcare, insurance, marketing, financial technology, and more.

2.2.2 Classifiers

Classification predictive modelling is the process of estimating a mapping function from input variables to its output variables. A classifier in ML automatically categorizes data into one or more classes. It is an algorithm itself defined by the rules used by machines to classify them. On the other hand the classification model is the classifiers product. The classifier is used to train the model, which ultimately categorizes new data, by utilizing training data to understand the relations of input variables to the class. Classifiers allow users to constantly tailor them to changing needs in order to adapt to new circumstances. There are several kinds of classifiers, in example: Support Vector Machine, Random Forest, Decision Tree, and finally the most famous one, artificial NNs. We differentiate between (semi-)supervised and unsupervised classifiers.

2.2.3 Semi-/ Un-/ Supervised Learning

In ML there are two basic approaches, supervised and unsupervised learning. A model in supervised ML learns from the labels of the training data over time. The prediction task is called a regression problem, if the result is a numerical measurement of quantity mapping $n \in \mathbb{N}$ inputs to a real valued output $f : \mathbb{R}^n \to \mathbb{R}$. Otherwise we refer to a classification problem. Then the function $f : \mathbb{R}^n \to \mathbb{N}_0$ maps to one of the $m \in \mathbb{N}_{\geq 2}$ different classes. The output is of a single natural number $i \in \{0, \dots, m-1\}$ indicating the classes index. Another common form is the one-hot encoding vector $\mathbf{y} = g(\mathbf{x})$, that is generated from the algorithm, given by the function $g : \mathbb{R}^n \to \mathbb{N}_0$, defined as follows:

$$y_i = \begin{cases} 1 : \text{if } x \text{ in class } i, \\ 0 : \text{else,} \end{cases} \tag{2.9}$$

Unsupervised models assess and discover hidden patterns without the need of human intervention, e.g. labeling. They are used for three main tasks: clustering, association and dimensionality reduction. Semi-supervised learning involves both a relatively small number of labeled and often a significantly larger amount of unlabeled examples. To make use of this mixture, specialized algorithms are required. It is particularly useful when you have a high volume of data and it is ideal for medical images where a small amount of training data can lead to a significant improvement in performance. Especially in case of a small amount of marked data, this method is of importance.

This work refers to subsequent classification as part of the supervised learning category.

2.2.4 Deep Learning

In DL algorithms are developed and modeled after the human brain. The development is traditionally a two-stage process. The first stage includes training and validation. In training phase the algorithms are applied using historical example data with known outcomes deriving and uncovering patterns between its features and the target variable to tune the models parameters developing the models value and leading to an overall performance improvement.

The second step is scoring, a.k.a. testing. Scoring is a key component of understanding ML model outcomes. A new data set from the same origin is applied to the previously trained model. The outcomes are in form of probability scores that indicate to what extend a sample belongs to a specific class, which can then be used for classification purposes. The final step is performed by a specific rule, e.g. a threshold or the highest probability for multi class classification. For every label X in a new data set assumed as true, there exists a prediction X by the model, which has been derived by the outcome variable from the model. Each value is assumed to be in $0, \dots, N$, where $N + 1$ is the total number of classes. In this work we concentrate on the most common case, a two class classification problem as opposed to the multi class classification problem.

2.3 Neural Networks

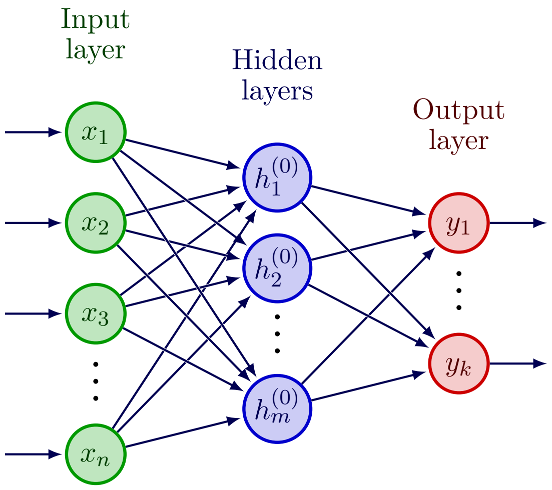


Fig. 2.3: NN connected to a layered system of neurons. Three layers connect the input variables from left (input layer with n neurons) via l hidden layers (for simplicity here only one) to output variables computed by k neurons in the output layer. [68]

A NN is similar to the artificial human nervous system. It is designed to receive information, process it and return the information in a desired format. It refers to a system of either synapses in the real world or neurons in an artificial sense. Multiple neurons in an artificial NN executed in parallel are aggregated into layer (see fig. 2.3 that may perform different transformations. Outputs from the previous layer are the inputs for the following attached layer. We differentiate between input layers, hidden layers of arbitrary multiplicity and output layers. The input layer takes given data in form of variables to be analyzed and forwards them through the hidden layers. Eventually the output layer combines previous variables to conclude a classification or regression result. The hidden layer is exclusively linked to other layers internally.

2.3.1 Neuron

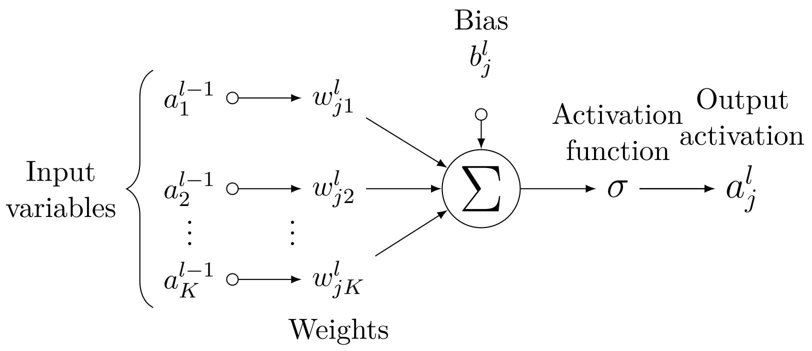


Fig. 2.4: Abstraction of a neuron with activation function applied to sum of weighted and biased variables. Variables are represented as small circles (variables $a_1^{l-1} - a_L^{l-1}$ left side and bias b_j^l at the top center), the activation output a_j^l on the right. [75]

In artificial NNs, a neuron forms a mathematical function (see fig. 2.4, that can be concatenated with others and is traditionally defined as follows. At first, all the real values a_k^{l-1} from the k^{th} input connection to the neuron are multiplied by their associated weight w_{jk}^l. L denotes the number of layers in the network and K the number of input connections. The specific bias for the j^{th} neuron in the l^{th} layer is then added to the weighted sum of all products resulting in the weighted input z_j^l. We finally apply an activation function σ to the weighted input. The overall output is also referred to as activation a_j^l. The bias provides each neuron in every node with a trainable constant b_j^l and allows to shift the activation function on the x-axis. Each connection has its own weight that can be adjusted together with the bias during training (see eq. (2.10)). We observe, that the activation for the l^{th} layer is related to all activations of the $(l-1)^{\text{th}}$ layer.

$$a_j^l = \sigma(z_j^l) = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right). \tag{2.10}$$

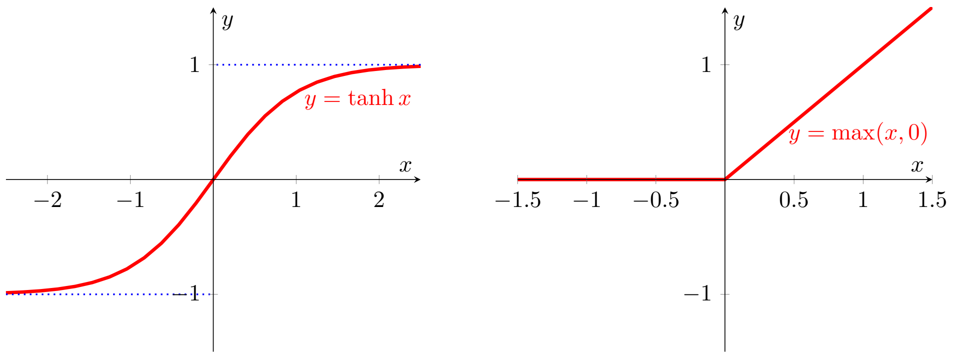
2.3.2 Activation Functions

An activation function basically decides whether the information given by its input values are relevant or not and activates or deactivates the neuron accordingly. The choice of the right activation function is essential for the application of a ML model. In practise many mathematical properties play a crucial role in real world application, influencing factors as e.g. computational costs or the ability of a model to approximate a special function based on given data. A back propagation algorithm traditionally uses gradient descent, that requires the derivative of the activation function to learn the weights and biases of a neural network from the gradient loss (see backpropagation section). For this reason the functions that can be used as an activation functions in NNs have to be monotonically increasing, differentiable and continuously defined everywhere.

The simplest kind of activation function is the linear activation, where no transformation is applied at all, but it does not allow the network to learn complex mappings. Similarly the binary step function with $f(x) = 1$ If $x > 0$ else 0 If $x < 0$ can be used for classification in regression. Next to that we have the linear activation function $y = m \times z$, where m is a constant variable. These approaches are not suitable to train complex NNs due to their constant derivatives and wide range of activations, which could blow up easily.

$$S(x) = \frac{L}{1 + e^{-k(x - x_0)}}, \text{ for } L = 1, k = 1, x_0 = 0 \tag{2.11}$$

Rather promising approaches are traditionally provided by the tanh and the sigmoid function eq. (2.11). The sigmoid s-curve used in the 1990s, is a special case of the logistic function and transforms input values from the real numbers to a range between zero and one. Later the tanh function turned out to produce slightly better performing models in praxis than the sigmoid function, even though it maps to a range between minus one and one. Sigmoid function and tanh are both continuous and differentiable. Unfavorably both methods suffer from the same phenomenons. They are only very sensitive around their midpoint of their input and they saturate for small and large values, meaning they snap to zero or one when converging to them. [40]



(a) Hyperbolic Tangent.

(b) Coordinate system illustrating rectified linear activation function.

Fig. 2.5: Line Plot comparison of Hyperbolic Tangent (a) and Rectified Linear Activation (b) for negative and positive inputs. For illustration purposes the Hyperbolic Tangent has been multiplied by a factor of four.

Most widely used piecewise linear function within hidden layers is referred to as rectified linear activation function (ReL) given as $f(x) = x$ If $x > 0$ else 0 If $x < 0$. The node implementing its function is the ReL unit, in short: ReLU. We note that the derivative of zero inputs is assumed as zero as well, even though it is strictly speaking not differentiable everywhere. Experience shows that gradient descent still performs well (see [40]). Because the rectifying function is linear for values greater than zero, it has a many desirable properties when training with backpropagation. The rectifier function is less computational expensive and sparse in representation compared to tanh and sigmoid function, allowing an accelerated us of NNs in routine development. A neural network with linear or close to linear activation functions are easier to train and avoid the vanishing gradient problem. This opens up the possibility to also train deeper networks and optimize hardware further. ReLUs therefore became the practical default activation function nowadays. To reduce generalization error we apply L1 or L2 penalty to the transformed activations, which hinders them to grow arbitrarily large. [38] ReLU also allows fragile (or "dying") gradients, leading to units never activating and learning with initial values of zero to be slow. Thus initial values of 0.1 or 1.0 are preferred, but they can not avoid this phenomenon completely. [65] Leaky ReLU defined similar to ReLU tries to tackle the problem of dying ReLU. It differs in that small negative values are allowed when input values are less that zero. It has a constant, small and negative slope for input values below zero. Nevertheless it lacks behind sigmoid and tanh when learning functions with high complexity. Next to other activation functions like Exponential Layer Unit (ELU), Parametric ReLU (PReLU) or Maxout it is one of the most common methods where we have challenges like sparse gradients often occuring generative adversarial networks (GANs).

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.12}$$

A softmax activation function, also know as normalized exponential function (see eq. (2.12)) is mainly used to form output layers for classification problems. The numerator is always the exponential function of the variable z_i) for which the corresponding softmax value is to be calculated. It sums up the exponential function of all z) components. As opposed to the sigmoid function it takes all other variables (z_i) from the previous layer into account and weighs them such that all outputs sum up to one. Note that the sigmoid is just a special case of the binary softmax function where all but one variables are set to zero. It has to be mentioned that the returned "probability" values have no real meaningful and informative value with regards to actual probabilities. Still, the transformation can be easily used in example for logistic regression models to build different layers of NNs outputting one-hot-encoded (see section on supervised learning) sequence with the help of argmax function. For binary models choosing two neurons (with softmax) in contrast to one neuron (with sigmoid) can be effective for training and classification, but may be superfluous.

2.3.3 Training and Backpropagation

In ML we differentiate between forward and backward Ppropagation. Forward propagation is referred to as feeding a network with data and computing the function that returns a prediction. Backward propagation (or backpropagation) on the other hand is the process of adjusting the weights and biases based on the difference between a prediction and the actual result.

Training begins with assigning the weights and biases. For initialisation there are mainly three options. One can initialize weights and biases with all zeros, which would serve almost no purpose, since we would face the problem of braking the symmetry [93]. Second we can simply initialize the model with random weights and biases. The third option is to take the weights and biases from a different pre-trained model. In this case the training phase can be subdivided into pre-training and fine-tuning the weights and biases with the actual data to reduce the error as much as possible. Pre-training can be performed on a different data set. The purpose of pre-training a network on a different data set first is a faster convergence to its optimal values and a better performance when fine-tuning on small data sets. In general, pre-training only makes sense when the data sets differentiate only slightly, e.g. both hold images of similar nature. The bigger the gap, the less effective it is. Often pre-trained networks are provided publicly for specific purposes and such pre-trained networks give us the initialisation values [95].

The model is applied to the data set and new weights and biases are then computed with the goal of reducing the error value across the training set to a minimum. Backpropagation describes how the weights and biases in a neural network change the cost function. A cost function is a technique or measure to evaluate how wrong a models performance is. There are several different commonly used cost functions available, e.g.: mean (absolute) squared error (MSE) loss, hinge embedding loss and negative-likelihood loss. One of the most popular cost functions refers to as the cross-entropy loss, where L denotes the total number of layers and N the number of training examples. We also define the corresponding desired output $y(x)$ as well as the activation a^l_j just like in the previous section about neurons. We average over the a sample set resulting in the cost function

$$C = -\frac{1}{N}\left(\sum_{i=1}^{N} y(x) \cdot \log(a^L(x))\right) \tag{2.13}$$

Cross-entropy is a better measure than MSE, because the decision boundary in a classification task is large. MSE is the right loss for regression, but does not punish misclassifications enough, because the distance between predicted values is small. The inner product is denoted by "$\cdot$". Further we assume that C can be computed by the average derivatives over all training examples C_x and it can be written as a function of the last layers outputs a^L. [69] [28]

For backpropagation we need to compute the partial derivatives of the cost function C with respect to the weights and bias first. The backpropagation algorithm can be described in five steps. In the first step we set the activations for the input layer. Next, we feedforward the values by computing the activations for the neurons in all following layers. In step three we compute the error vector δ^L for the output layer.

$$\delta^L = \nabla_a C \odot \sigma'(z^L), \tag{2.14}$$

The error is denoted as δ^i_j for the i^{th} neuron in the j^{th} layer defined as $\partial C/\partial z^l_j$. Here $\nabla_a C$ is defined by the vector of partial derivatives of C with respect to a^L_j. The two terms on the right multiplied by the Hadamard product measure "how fast the cost is changing as a function of the i^{th} output activation" multiplied by how fast "the activation function σ is changing at z^L". [69] In the fourth step we compute the error of previous layers $l = L-1, L-2; \ldots, 2$ in terms of the error of layer L as follows:

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l). \tag{2.15}$$

By repeatedly applying this equation to previous we propagate the error backwards through the network and get errors for every weight. This is possible due to the chainrule. Lastly we compute the rate of change to the costs in the network with respect to any bias with: $\delta^l_j = \partial C/\partial b^l_j$.

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j. \tag{2.16}$$

In addition to that we compute the partial derivatives with respect to the weights w^l_{jk} (left side of the equation) by multiplying the error of the neuron output with the activations of the neurons input (see eq. (2.16)). [69]

Practically the errors are computed for sample batches of different sizes, then averaged and applied all together. The batch size has an impact on the models performance. For a higher batch size the error fluctuates less, but also fails to put rarely occuring features into account. Most common and effective batch sizes in image classification are two to the power of 4 to 10. [54] In the following optimization algorithms or optimizers iteratively try to find a local minimum of a function. For this it is assumed that the function is convex and differentiable. Stochastic gradient descent is often preferred over batch and mini batch gradient descent, because it is faster and is able to take rare features into account. The Adam optimizer solves issues of the gradient descent algorithms, like finding the right learning rate, dealing with rare features in sparse data and getting stuck in suboptimal local minima. As one of the class of adaptive optimizers it automatically tunes the learning rate value while training. Generally, the Adam optimizer performs best for CNNs [103]. In ML the optimizers adjust the learning rate before calculating new weights as follows:

$$\mathrm{NC} = \mathrm{LR} * \mathrm{E} + \mathrm{M} * \mathrm{LC}\,. \tag{2.17}$$

This process makes increases training efficiency significantly. The learning rate (LR) controls the speed of the models learning process. Then, the optimizer reduces the learning rate, which is dependent on the calculated gradient. An optimal initial value often lies in the order of a few digits after the decimal point. The training process generally starts with a high learning rate which is reduced in the process. Error (E represents the difference between what the output with specific parameters in the current model is and what it should be. [31] High learning rate can lead to an oscillation and prevent the model from converging to an optimal solution. A lower rate may result in an unnecessary long training time. The gradient descent could get caught in a suboptimal local minimum (see fig. A.3). There may be multiple optimal solutions as local minima. The momentum (M) prevents widely fluctuating new weight changes (NC) by multiplication with last weight change (LC). [31] [12]

2.3.4 Residual Networks

As a rule of thumb the performance of a NN increases the more layers we stack to each other. The general intuition behind a NNs as function approximators suggests that more layers increase their overall performance. But this generalisation can be misleading, since “the deeper the better” does not always hold. In 2015 Kaiming He and colleagues have shown that performance saturates degrades after some depth due to vanishing gradient (like in VGG nets [85]) and other related phenomenons [44]. This degradation problem results in weights and biases never updating its values, hindering the CNN to learn simple functions e.g. the identity function without further ado. Problems like this can be minimized by bypassing layers in between with a residual connection introduced in the context of information highway networks in 2015 [88]. This way larger gradients can be propagated directly through the skipping connections, a.k.a. identity shortcut connections (see fig. 2.6). Thus we have $H(x) = R(x) + x$ such that the network learns the residual $R(x)$ decisively faster as opposed to a traditional network learning $H(x)$ [44]. This allows us to train much deeper and better performing networks than before, e.g. with AlexNet [59].

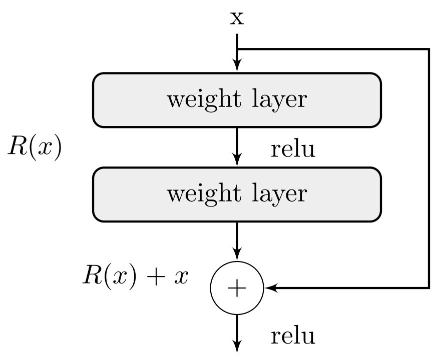


Fig. 2.6: Single Residual Block. [44][76]

In ResNets we can traditionally observe residual blocks with either two or three weight layers that are skipped by a shortcut connection defining a single residual block. It has been shown that more dense shortcuts allow even deeper NNs and therefore there exist residual blocks with only one weight layer.[49]. There exist different versions of weight layers. Pre-activation layers with first a batch normalization and the subsequent activation function mostly give the best results [45], but other sequences are possible as well. From an intuitive perspective, we can see the skip connections as dynamic hyper-parameters that are tuned automatically. They determine how many layers are optimally used and which nodes are used during training. There are different versions of ResNets, but their general architectural concept remains the same.

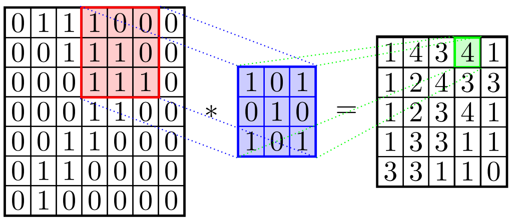


Fig. 2.7: Convolution using Cross Product. [100]

One of the fundamental concepts of a ResNet are convolutions layers. Having convolutional layers makes the ResNet a special type of CNN. The basics of a CNN architecture consist of a convolution, pooling, and fully connected layer. In contrast to fully connected layers they look at specific features instead and use filters (or kernels) to extract specific features. This has several reasons. Among them are computational efficiency and provable performance improvements, e.g. with regards to training speed and prediction performance. [40] The two dimensional filter is multiplied with a filter-sized patch from the input data. This scalar product results in a single valued score. A convolutional layer then applies multiple overlapping convolutions systematically across the input volume returning a feature map, on which an activation function can be applied. Note that convolution here may in other contexts be referred to as flipped cross-correlation. [40]

The number in the name of a specific ResNet (e.g. ResNet-18) denotes how many layers there are. Commonly a ResNet begins with a 7x7 convolution with a feature map of size 64, batch normalization (elementwise) and max-pooling operation. The 64 refers to the number of kernels (or feature maps) that are convolved with the input. With this techniques the model becomes the ability to recognize specific features from a network that has learned rich feature representations for a wide range of images. batch normalization is a technique that standardizes inputs to a layer batch wise. Training speed can be drastically increased, when applying it after each convolution. [13] Max-pooling refers to calculating the maximum value for each patch of the feature map. The convolution is padded with three zeros and the max-pooling operation has a stride of two. Stride is how far the filter moves forward in every iteration step across the data. All together this halves the input dimensions from (224) to a (122) volume. Note that for simplicity this is free of multi batch dimensionality, e.g. thee channels in one image. This is often used for colorized images, but does not change general principles. The following layers (see e.g. tab. B.1) are composed of several basic blocks with two operations (see fig. 2.6) as opposed to bottleneck blocks with three operations. [44] Note that these operations were previously also referred to as layers. The layers here consist of multiple blocks, most of which do not change the volume size. Unlike in normal CNNs, where downsampling is done by pooling operations, the size is changed by increasing the stride of only the first operation in the layer from one to two. When size changes we use projection shortcuts with a convolutional operation instead of the usual identity shortcuts. Duplicating the number of filters, when downsampling preserves time complexity. An average pooling layer and a fully connected layer are appended at the end of the ResNet. An average pooling layer calculates the mean of each feature map before concatenating all the values into a one-dimensional (1D) list. [44]

2.3.5 Metrics

While the network evolves as well as after its development process, monitoring of processing elements, values and overall performance enables us to make fundamental decisions about parameters like its training time, overall topology and learning parameters. One way to make investigations on ML models based on observations is to create a heat map. When doing a sufficient number of tests for one model, as described below (see estimation of errors), we can compare different models and choose the most accurate one that produces the valuable insights based on computed metrics.

Performance metrics are extremely useful when evaluating and comparing the different classification models or ML techniques resulting after training. To test the ability of a so called classifier, there are different metrics that prove useful for comparing the performance of two different models and analyzing the behavior of the same model by varying different parameters. In this work we concentrate on classification metrics as opposed to regression problems, where relative errors play a more essential role.

Subsequently condition positive (P) represents the number of real positive cases in the data, whereas condition negative (N) stands for the number of real negative cases in the data. We used the following metrics to assess the segmentation performance of our models.

Confusion Matrix

The 2D confusion matrix (in unsupervised learning a.k.a. matching matrix) compares the instances in actual classes (here rows) with the instances in predicted classes (here columns) (tab. 2.1), so that for C_s as either the predicted or actual sum of classes holds:

$$C_s = P + N \tag{2.18}$$

As a special kind of contingency matrix it easily illustrates weather two classes are commonly interchanged with each other. We make the correct results appear in green for easy identification. Confusion matrices can in general also be used as multi-class classifiers as well [42], where the diagonalized matrix at best represents a unit matrix. We will concentrate on binary classification in the following though. In total there are four types of basic confusion matrix rates. True positive (TP) results that correctly indicate the presence of a condition or characteristic whereas true negatives (TN) correctly indicate their absence. Corresponding to this, false positive (FP) results wrongly indicate that a particular condition or attribute is present and false negatives (FN) wrongly indicate their absence. The sum of all fields equals the total number of samples. The so-called

Tab. 2.1: Confusion Matrix

		Predicted condition	
		Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

confusion matrix allows a much more comprehensive analysis of a binary predictive analysis. Hereafter, we will discuss these types of analysis in detail for bi-classification systems.

Classification Accuracy

Calculating the classification accuracy (ACC) (or weighted ACC) is the most obvious and common technique to investigate the proportion of correct classification in a confusion matrix. We compute is by taking the arithmetic mean of all True (1) and False (0) Predictions, see eq. (2.19). In most diagnostic cases of diseases it reduces to dividing the sum of true by the total number of classifications. To its inconvenience it yields misleading results for unbalanced data sets, because it gives more weight to instances coming from the majority class.

$$\begin{gathered}\mathrm{ACC} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}(x_1 + \cdots + x_n) = \frac{TP + TN}{TP + TN + FP + FN}\ , \\ \text{for } x_i \in \{\text{all True (1) and False (0) predictions}\}\end{gathered} \tag{2.19}$$

Sensitivity and Specificity

Sensitivity (SENS) or Recall refers to the probability of a positive test, conditioned on truly being positive, see eq. (2.20) [107]. It is a measure of how well a test can identify true positives.

$$\mathrm{SENS} = \frac{TP}{\text{Actual True}} = \frac{TP}{TP + FN} \tag{2.20}$$

Specificity (SPEC) (or selectivity) refers to the probability of a negative test, conditioned on truly being negative (eq. (2.21)) [107]. It is a measure of how well a test can identify true negatives. SPEC can be calculated according to the negative scheme as SENS

$$\text{SPEC} = \frac{TN}{\text{Actual False}} = \frac{TN}{TN + FP} \tag{2.21}$$

Higher SENS generally means lower SPEC and vice versa, which is why there is usually a trade-off between them. To identify a true condition accurately the SENS should be high. To identify a false condition accurately the SPEC should be high, that is 100% in an optimal case. SENS and SPEC are intrinsic measures and therefore independent of the prevalence. Because SPEC does not consider false negatives, a negative result from a test with high SPEC is not necessarily useful for ruling out disease. The same applies to SENS, which does not include false positives and therefore should not be used to detect a disease. Therefore the diagnostic power of any test is usually determined by both, its SENS and its SPEC [11].

Miss rate refers to the SENS's and fall-out to the SENS's complement. Because SENS and SPEC measures are correlating, a case-specific weighting must often be determined. This trade-off is explored in receiver operating characteristic (ROC) analysis not considered in detail.

Balanced Accuracy

By computing the arithmetic mean of SENS and SPEC, we get the balanced ACC (BACC), see eq. (2.22). It offers the advantage that it does not suffer from unbalanced data sets.

$$\text{BACC} = \frac{\text{SENS} + \text{SPEC}}{2} \tag{2.22}$$

Positive and Negative Predictive Values

Positive and negative predictive values (PPV and NPV respectively) are the proportions of positive and negative results in diagnostic tests per true positives and true negatives [86]. For instance PPV is often also called precision (eq. (2.23)), due to its statistical relevance. NPV can be computed following the same pattern. A perfect test result would have a score of 1 (for 100%), while the worst test would give a score of 0.

$$\text{PREC} = \frac{TP}{\text{Total True Predictions}} = \frac{TP}{TP + FP} \tag{2.23}$$

PPV and NPV are not intrinsic and therefore depend on the prevalence (overall proportion of positive test results). That means a higher prevalence results in a higher PPV and conversely a lower prevalence in a lower PPV. The opposite is true for NPV. To meaningfully compare multiple test sets an equivalent prevalence is therefore inevitable. This is often achieved by normalizing the prevalence to 50%. We have the false discovery rate (FDR) and false omission rate (FOR) as complements to PPV and NPV.

F1-score

The F1-score ($\beta = 1$) or dice coefficient (after its geometric interpretation) is a special case for the F_β measure defined in eq. (2.24). It is the harmonic mean of PPV and SENS. In contrast to the BACC the F1-score doesn't take into account how many true negatives are being classified or the number of negative examples at all. "It generates a high score only if the number of true positives obtained is high compared to the other confusion matrix cells." [15] Therefore we only prefer F1-score over BACC for imbalanced data when more attention is needed on the positives. It highly depends on which class is defined as positive. The F-measures key advantage is that it can apply additional weighting, valuing either PPV or SENS more than the other. The lowest possible score is 0 if PPV or SENS is zero, while the perfect value is 1 indicating a perfect classification of the system. The F-measure by Van Rijsbergen was derived so that F_β "measures the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision" [102].

$$\begin{gathered} F_\beta = (1+\beta^2) \times \frac{PPV \times \text{SENS}}{\beta^2 \times PPV + \text{SENS}} \\ \Rightarrow F_1 = \frac{2(PPV \times \text{SENS})}{PPV + \text{SENS}} = \frac{TP}{TP + \frac{1}{2}FP + FN} = \frac{2TP}{2TP + TP + FN} \end{gathered} \tag{2.24}$$

Matthews Correlation Coefficient

Another balanced measure to evaluate the classification performance is denoted as the phi coefficient (ϕ) or MCC. It describes the correlation between the observed and predicted classifications and can be calculated from the confusion matrix eq. (2.25) as well. A coefficient of one indicates a perfect prediction, zero is no better than a random prediction and minus one indicates the worst prediction possible. It does not yield a reliable estimate on how close the classification model is to a random prediction. The MCC takes into account the balance ratios (more precisely: low precision) of the confusion matrix entries and unlike the F1-score doesn't depend on which class is the positive one, which is why it is preferred for balanced data sets. To its detriment, MCC deteriorates decisively with unbalanced data sets [110].

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{2.25}$$

General Remarks

"Even if four-category confusion matrices are a common tool to evaluate binary classifications in supervised ML and statistics, no general consensus has been reached on a unique statistical score able to informatively recap its key-message." [19] Other common metrics like diagnostic odds ratio (DOR), Jaccard coefficient, BACC weighted [42], bookmaker informedness and markedness [20] do not suit for evaluating a binary DL classification system or are less informative than the measures described above and will therefore not be considered further in this study. For instance diagnostic odds ratio (DOR) does not take into account precision and negative predictive value [19]. As the main disadvantage Fowlkes-Mallows index does not consider true negatives. In this work we do not consider BACC weighted that keeps track of the importance of each class for imbalanced data sets. Research has also shown that the kappa measure should be avoided over MCC [24]. For this reason, we focus on the most promising and common metrics in order to get more meaningful results as well as maintain consistency and practicality for the purpose of information transfer. Confidence intervals can be used to evaluate metrics and see how precisely we can measure our metrics performance. These methods have limited meaningfulness though, when data samples are correlated to each other and training sets are small.

In DL, the rule of thumb is that more data results in better performance and vice versa. Small data sets often limit the training capabilities and the resulting performance of trained. Another factor that significantly affects the performance of a DL model is the quality of the data used to train it. This includes various factors such as the information content, the diversity, the independence and the preciseness of the manually determined variables (or labels). Collecting more training data is often a challenging task or not feasible for medical purposes. This is due to high costs or the limited possibilities to create or collect sample data. Also, labeling large amounts of data often involves an immense effort or cost, not to mention the professional expertise and certain knowledge that is required to label data consistently. These issues arise particularly frequently in the medical field. In order to mitigate these shortcomings and still achieve a valuable performance of the DL models, DA is applied.

2.3.6 Overfitting and Generalization

The most important thing to do while and after training is to ensure as well as evaluate the classifiers applicability. One of the main goals is to prevent or delimit overfitting (or variance) to its minimum. We likewise want to avoid underfitting (or bias) and achieve a certain level of generalizability considering that the model should be similarly well applicable to data from the same domain. Overfitting is "a modeling error in statistics that occurs when a function is too closely aligned to a limited set of data points. As a result, the model is useful in reference only to its initial data set, and not to any other data sets." [97] In ML it refers to high error with respect to test data as well as low prediction error with respect to training data. Overfitting occurs when a model learns details and noise of the training data that are not reflected in the test data it is applied to later, e.g. when testing fig. A.1.

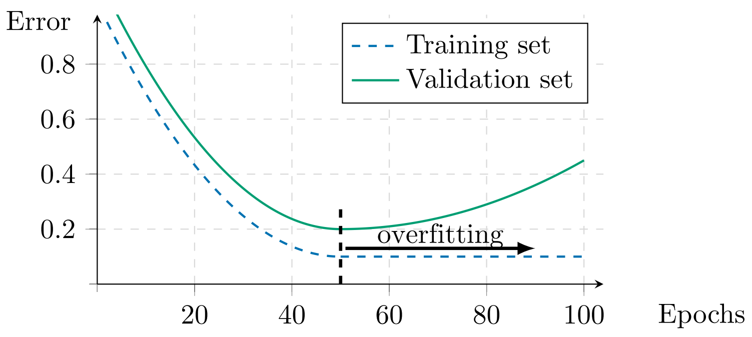


Fig. 2.8: Smoothed line plot of training error (y-axis) for every epoch (x-axis), highlighting the sweet-spot from which on model performance decreases due to overfitting. [99]

It can be recognized by comparing the models performance with seen and unseen data. When overfitting occurs in the training process, there may be a spot from which on error on unseen data becomes worse than before while the seen data perform distinctively better than unseen data. The goal is to create a model that generalizes well fig. 2.8. Generalization refers to how accurate the model performs on unseen examples compared to the ones it has been trained with. Therefore, the one goal is to find this sweet spot.

The easiest way to prevent overfitting while training is to stop the training process when the error rate on the validation set increases. But there are several other techniques presented and used in the following work to diminish over- and underfitting while training to obtain more representative performance values.

Pruning and Dropout

Overfitting can be promoted due to an overly complex model with too many parameters. Pruning refers to the process of removing entire groupings or layers of nodes and weight connections from the neural network to increase inference speed and decrease model storage size, while affecting metrics such as ACC as little as possible. We differentiate between structured and unstructured pruning. Removing layers is a simple way of pruning. A more complex example is provided by algorithms removing nodes and connections using specific criteria that maintain the performance of the model as good as possible. Furthermore dropout describes a variant that randomly deletes single nodes with a certain probability from all layers except input and output layers. In each epoch a different set of nodes is kept so that the neural network can not rely on any input node. Thus it will become reluctant to give high weights and biases to certain features and a small quantity of nodes, because it is random if they disappear. One can remove features from the training data to reduce the risk of overfitting, but this is not feasible for a binary classification.

Regularization

$$\text{Loss} = \text{Error}(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i^2| \tag{2.26}$$

L1 and L2 regularization (after L1, L2 norm, a.k.a. lasso regression and ridge regression) are two other possibilities to mitigate overfitting. Ridge regression adds the sum of squared parameters w_i multiplied by the coefficient λ to the loss function, which penalizes large weights and biases of a model in order to minimize the cost function (see eq. (2.26)). The loss function here predicts the difference between true values λ and predicted values $\hat{\lambda}$. The λ value is determined using cross-validation. Computationally, lasso regression only differentiates in that it does not square the variables, but takes the absolute value instead. With this it can shrink and exclude useless variables from equations completely to zero.

Cross-Validation

CV or rotation estimation is the model validation technique assessing how well a model generalizes to predict new labels from unknown and independent data sets with the same origin. There are several different variants of exhaustive and non-exhaustive CV, e.g. leave-one-out or K-fold CV. Non-Exhaustive techniques do not compute all ways of splitting the original sample, mainly to limit its complexity for a rising number of data, while exhaustive ones do. To be able to train a model and at the same time meaningfully measure its performance, a data set is usually divided into three empirically separated subsets: training set, validation set and test set (or holdout). K-fold CV uses different subsets of the data to train and test a model on multiple iterations. This enables a quite accurate estimation insight of the algorithms performance quality indicating characteristics like selection bias or overfitting. The training set together with the validation set (known labels) make up the main part of a data used, which is usually around 70-90% of the entire data set. Test data (unknown labels) make up the rest. The validation set size is traditionally in the range from 1/5 to 1/20 of the test set. Both sets are rotating in every CV-step used during the training process of the algorithm to calculate its performance measure (see fig. A.2). Unknown test data remain static and unchanged. In every CV-step we train a new model with the specific validation holdout, which gives us n trained models. Performance is then assessed in the scoring phase where each model is applied to the test set, such that measures can be computed from the metrics (see metrics section) independently from training. The described variant of K-fold CV fulfills its purpose relatively well and at the same time keeps the computing costs within a realistically applicable range. Taking into account that estimates for small K are usually more noisy and have an upwards bias, larger k values are desirable. Because the computational burden and the leave-one-out CV is too heavy, a K-value equal to five or ten may be used. [33] Distinguishing between test set and validation set prevents each model from overfitting on the data it has been validated with, which leads to a decisively more accurate testing result. In the end we average all score variables creating a summarizing performance indicator representative for a model trained with the entire training set.

2.4 Data Augmentation

In DL, the rule of thumb is that an increased amount of data results in better performance and vice versa. Unlike in the natural image domain, there are far fewer training images available in various other domains. Small data sets often limit the training capabilities and the resulting performance of possibly high value models. Many parameters are undetermined in low-data regimes, which leads to poor generalisation of learnt networks. Collecting and labeling training data happens to be one of the greatest challenges when creating qualitative DL models. Particularly in the medical sector, there is often a lack of professionally labeled data and anatomical images are typically very limited. Due to the immense effort and costs, collecting large data sets is often be unfeasible. Another factor that significantly affects the performance of a DL model is the quality of the data used to train it. This includes various factors such as the information content, the diversity, the independence and the preciseness of the manually determined variables (or labels). Since we often have very limited to no control over these factors in medical low-data regimes, DA became an essential part of training CNNs. In order to make simulation feasible and simple, Wong and Tanner popularized DA initially [91]. DA uses existing data with increased effectiveness to overcome sparsity of training data. It refers to the application and the combination of various DA techniques to the corresponding training data to expand as well as diversify the original training data set. DA is applied in order to mitigate the above described shortcomings and still achieve a valuable performance while diminishing overfitting of the DL models by changing the images pixel values but keeping the majority of information in it. It is aimed at increasing the generalization of the resulting DL model and thus achieving a more reliable and ultimately a higher overall prediction performance. DA techniques and their combination have been heavily researched for natural image tasks. Even if DA is widely utilized in medical imaging for DL, little research has been done to determine which augmentation algorithms capture

IVOCT image features best, producing strategies for more discriminative models [50]. In the following, we focus on different DA techniques that have publicly been proposed at an earlier point of time to transform IVOCT images. For the sake of this study we differentiate between commonly applied DA techniques and (intravascular) OCT specific approaches.

2.4.1 State of the Art

Intuitively common DA techniques can not always be usefully applied to polar as well as Cartesian formats the same way. Different approaches for transformations can be differentiated and therefore grouped by their individual characteristics which have distinguishing effects on the images they are applied on. We group the approaches into three categories. To the first category belong approaches that are usefully applicable to the polar format, then the Cartesian format and finally we group applicable to both formats. Because IVOCT images are grayscaled we restrict ourselves to filters applicable to images with a single channel.

Affine Transformations

Most common approaches on polar representations are shifting [61, 56] and random flipping. For the former approach the images are shifted randomly along the θ direction with $s_\theta \in [0px, 300px]$. A-scans are consequently shifted out along the positive direction and must be added back to the opposite site. Random horizontal flipping can additionally be applied along the temporal axis in θ direction [25, 34, 56, 35, 36]. Overlapping A-scans need to be appended accordingly. In combination one can observe the commutative property of those random transformations. As a result, one only randomises in either shifting or flipping when combining them, as the output set and probability distribution stays the same.

In Cartesian representation images are often rotated randomly [8, 25, 61, 50, 56, 35, 36] with $\theta \in [0°, 360°]$ which can cause information loss due to interpolation for $\theta \notin (0°, 90°, 180°, 270°)$. Random flipping [25] can be applied to Cartesian images in every angle due to its circular nature (undefined edges values remain static). Rotation around the center has an almost equivalent effect on the outcome, as if we apply a random shift in polar representation and convert it to a Cartesian representation afterwards. The main difference is, that it is restricted to a an angle of $2\pi/\theta$, such that an extra interpolation can be avoided. This is why one can neglect center rotation, when the mapping between coordinate systems takes place. Random flipping and rotation are not commutative. If interpolation is neglected, the set with possible outputs also differs, if one of the two random factors is limited to a certain range of possible values. Flipping along the x and y axis is least cost efficient, because interpolation can be neglected. One can produce approximately the same result set and probability distribution by applying the random flipping on x and y axis first and a rotation in a second step. A polar image may be rotated by an interpolation after applying a mapping described by the matrix A, where θ is the rotation angle:

$$A = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \text{ where } \theta \text{ is the rotation angle.} \tag{2.27}$$

In contrast to central flipping and rotation, other affine transformations [37], specifically magnification adjustments [61] can change the images shape. One way is to crop (or trim) the image randomly [35]. Another option is to pad the border with either zero values, by repeating the outer most values, or by mirroring the image in the according direction. A special case of this magnification adjustments are scale adjustments, that is, zooming in (center cropping) [8] or zooming out. Assuming the origin in the middle of the image, zooming is multiplying coordinates by a constant factor with subsequent interpolation. If only one dimension is scaled we refer to it as stretching. Changing the aspect ratio of an image often means information loss and that it has to be interpolated. Kolluru applies cropping only for deeper pixels, because of the limited tissue penetration depth with IVOCT [57]. Note, that if we combine cropping and padding, we construct a new way of augmenting images, that scales and moves them in the mask. When we decide to crop the same amount as we pad on the opposite side, one often refers to image translation [8]. Finally, shearing can be used to shift one part of the image to a direction and the other part to the opposite direction [50].

Elastic Deformation

Elastic deformation [25, 84, 105] is a warping technique combining shearing and stretching and is often only applied to smaller portions of images. This makes our network invariant to images with atypical morphology. Its goal is to make the network invariant to samples with atypical morphology. Simard showed greater training improvement when training with elastic transformations instead of classical affine transformations [84]. Devalla defined a normalized random displacement field u representing the random unit displacement vector (values between minus and plus one) for each pixel location in the image as

$$P_w = P_o + \alpha u \tag{2.28}$$

where P_w and P_o are the pixel locations in the warped and the original image. The magnitude of the displacement was controlled by α and the variation in displacement by σ. After the deformation mapping, bilinear (or bicubic and spline) interpolation takes place. [84]

Intensity Shifting

Intensity shifts constitute an attempt to mimic the natural noise with a specific distribution and intensity for better generalization of the model. Single pixel intensity shifts are among the simplest and most often used DA techniques available. First we present inversion of intensities proposed by Kuwayama [61], which is given by a simple bitwise inversion for unsigned integers. Basic brightness and contrast [61] adjustments can be jointly described parallel with the OpenCV documentations [22] in a single formula

$$f\colon [0, \max] \longrightarrow [0, \max],\ f(x) = a(P_i - h) + h + b \tag{2.29}$$

Here p_{ij} is the pixel value in column i and row j of the image represented by a grid of intensities. A value of $a > 1$ increases the contrast (gain), whereas a scalar of $|a| < 1$ leads to a decrease. When a is equal to one, $f(x)$ equals the identity function such that pixel values remain the same. The b variable (bias) controls the brightness. Note that h is defined as $h = (\max + 1)/2$. Returned values always stay within the boundaries of [0, max], where max is the highest possible value representable, e.g. 255 for an 8 bit unsigned integer type. Brightness raises or lowers the entire range of tones within the image accordingly. When contrast adjustment is raised the image will have a higher percentage of white and dark pixels and eliminate middle tones. Contrast and brightness can also be applied in a jittering way [50], e.g. Hussain uses a small amount of contrast (+/- one to four intensity values)

Another augmentation or pre-transformation technique is to apply a pixelwise noise to the image. There are different variants of noise that can be applied. Gaussian noise [61, 50] is a term from signal processing theory. The signal noise has a probability density function equal to that of a Gaussian distribution. In other words, the values are Gaussian distributed. Hussain uses a variance between 0.1 and 0.9 for this. Additive noise values can also be generated randomly [61] in the range of a certain intensity limit. It may only be applied to a certain proportion of pixels, e.g. 5%. Devella and Kermani make extensive use of additive white noise, where intensities are exclusively increased but not decreased. [25, 55] In image processing salt-and-pepper noise is referred to as randomly changing a chosen proportion of pixels to white or black (its complete maximum or minimum). In some cases noise is added uniformly (speckle noise), but other options are to only add white pixels, or only add black pixels (salt or pepper). In signal processing often multiplicative noise (a.k.a. multiplicative speckle noise) [25, 37, 55] is signal dependent and difficult to remove. [17] Thus one can add multiplicative noise manually in an attempt to make the DL model learn to deal with such inconsistencies. Due to the coherent nature of image acquisition processes, the standard additive noise, so prevalent in image processing, is inadequate [17]. Multiplicative noise is dependent on the image data and an accurate description of coherent imaging systems is provided by multiplicative noise models, in which the noise field is multiplied with the original image [10]. Note that many filtering methods work well when the multiplicative noise is weak [17]. Therefore artificial speckle noise is utilized to make the DL models invariant to its presence.

Non-linear intensity shifts constitute a more complex and expensive technique [62, 25]. According to Devalla [25], these can make NNs invariant to intensity inhomogeneity within and between tissue layers in OCT imaging as follows:

$$\bar{I}_a = -a + (a + b + 1) \times I^c. \tag{2.30}$$

a and b are random numbers between 0 and 0.1, whereas the exponential factor c lies randomly between 0.6 and 1.4. The transformation is applied to the image intensities I pixelwise for the i'th column and j'th row, returning the modified image $\bar{I}_a$.

Convolutional Filters

In contrast to adding artificial noise, one can also use convolutional filters to remove noise and enhance images. Often unwanted details can be removed this way. The simplest convolutional filter is the mean filter, that means, every value in the filter map is equal to one. It take the average over all pixels in holds. The number of averaged pixels depends on the kernel size. Compared to the mean filter that distorts the image relatively much, the Gaussian filter [57] is closer to the original image. In a 2D Gaussian kernel the pixel values correspond to the Gaussian curve and add up to one fig. 2.9. The standard deviation can additionally be chosen by a variable, e.g. Kulluru uses a kernel size of 7x7 value of one. Third, the median [56] filter is particularly suitable for removing salt and pepper noise. Note that it is not described by convolution, but the values in the neighborhood covered by the mask are sorted and the median, (middle value) is returned

$$\frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}\qquad \frac{1}{273}\begin{bmatrix}1&4&7&4&1\\4&16&26&16&4\\7&26&41&26&7\\4&16&26&16&4\\1&4&7&4&1\end{bmatrix}\qquad \frac{1}{1003}\begin{bmatrix}0&0&1&2&1&0&0\\0&3&13&22&13&3&0\\1&13&59&97&59&13&1\\2&22&97&159&97&22&2\\1&13&59&97&59&13&1\\0&3&13&22&13&3&0\\0&0&1&2&1&0&0\end{bmatrix}$$

Fig. 2.9: An 3x3, 5x5 and 7x7 kernel for Gaussian convolution. [83]

as value. Compared to the Gaussian filter (or Gaussian blur), the median filter gives better results in less time [60].

Partial Masking

Another option to prevent overfitting is partial masking, where portions of the images are randomly erased. These cutouts may vary in size and shape. They are commonly used in form of simple squares with a random size, value and ratio between 1:1 and 1:3. Alternatively parts of the images can be replaced or swapped. An advanced level of masking is occlusion patching, which has been used by Devalla in the domain of IVOCT imaging as well [25]. The occluding patches reduced visibility of certain tissues, making our network invariant to blood vessel shadows. Davella used twenty occluding patches at random positions with a size of 60×20 pixels. The intensity of the entire occluded region was reduced by a factor between 0.2 and 0.8.

3 Methodology

3.1 Overview

In order to perform our DA experiments on IVOCT data, certain preparations must be made beforehand. We present a pipeline interpretation with a CNN, which allows us to better generalize observations from our experiments that are performed subsequently. In order for the IVOCT images to be processed well by the pipeline, we provide a detailed description on how data are pre-processed. Furthermore, new DA techniques and the conducted experiments are presented. The experiments focus on the behaviour of DA techniques applied in the domain of IVOCT data.

3.2 Implementation

Training of NNs is the most resource insensitive tasks in DL. Often the time it takes to train an model is very crucial. New weights have to be recalculated and updated hundreds or thousands of times. Sometimes millions or billions of weights have to be adjusted in each training step. This may take a huge amount of time if calculations are not optimized well enough. Training of a NNs essentially boils down to a huge number of matrix multiplications that can be parallelized to a high extend. Thus NNs can be accelerated by optimizing either software or the hardware components. Most importantly, both have to be coordinated accordingly.

GPUs as one of the class of Multiple Instructions, Multiple Data (MIMD) computing architectures excel at parallel programming. Thus programs that can be parallelized may be speed up significantly. For our experiments, we use special hardware to run the tests in feasible time as well. Our hardware setup consists of a 64 Bit Quad Core Intel Xeon E3-1225 v3 Processor at its core, that enables real parallel execution of multiple processes. We used 16 GB of Random Access Memory (RAM) and attached an 840 EVO 250GB SSD Drive for local storage. For training acceleration we use an NVIDIA GeForce GTX 1080 Ti graphics card with 8 GB of GDDR5X memory. In DL most highly optimized software is developed for the operating system we installed refered to as Ubuntu 20.04.4 LTS, a free and open source Linux distribution based on Debian.

3.2.1 PyTorch

In the field of DL there are specialized frameworks, that can exploit the full potential of multiple attached GPUs. Among others, TensorFlow and Scikit-learn, PyTorch are some of the most widely used DL frameworks for python [72, 73, 26]. PyTorch has been developed by Facebook's AI Research laboratory and was released in 2016 as a free and open-source python library. Its mainly used for computer vision, natural language processing and DL. While these frameworks differentiate in efficiency and usability, PyTorch excels at combining simple and imperative programming with its highly efficient implementation using hardware accelerators. This enables a more feasible usage with an easy scalability. PyTorch performs parallel, immediate execution of dynamic tensor computations with GPU acceleration. With a C++ core it offers complex functionalities, e.g. automatic differentiation for backpropagation, while maintaining performance with help of multiprocessing. Its performance is comparable to the fastest current DL libraries [72].

PyTorch helps us to easily implement our training pipeline to conduct detailed tests on DA. It can be subdivided into three important parts. First, we preprocess the given data, then we train on them and finally we evaluate the models performance using the metrics introduced above. Preprocessing transformations are fundamental dependent on the model used. Thus we present the model for our experiments first.

3.2.2 ResNet-18

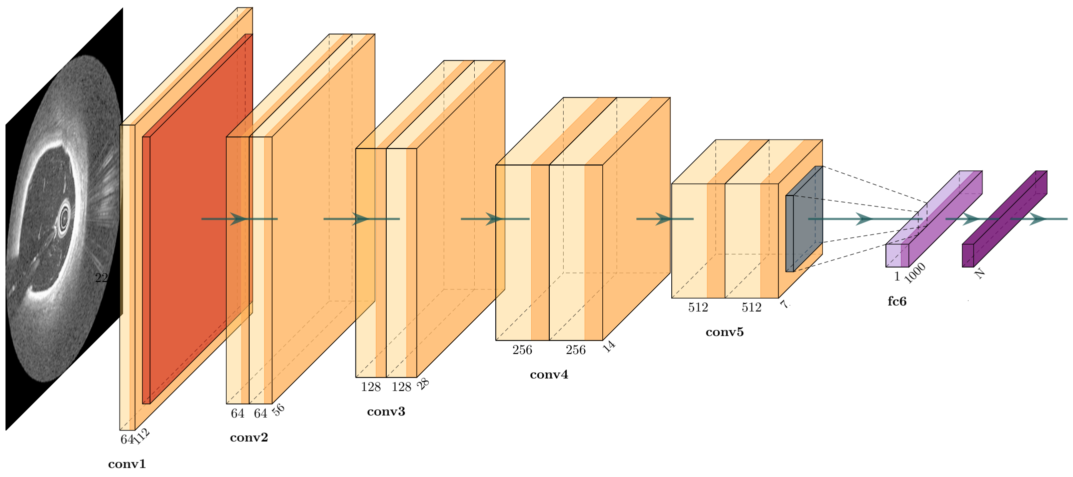


Fig. 3.1: ResNet-18 pipeline architecture with 5 convolutional layers (yellow) stacked directly on each other with a max pooling convolutional layer (orange) on the left and an average pooling layer (blue) on the right. It results into a fully connected layer (purple light) and a softmax function layer (purple dark) with output size of n variables. [51]

Using a widely and academically meaningful architecture that is comparatively well known and understood offers several advantages. First it gives us the opportunity to use DL frameworks that offer a variety of advantages and it allows us to fall back on existing functionalities and data like pretrained models. Lastly, well known models allow for a deeper level of interpretation than others. For image classification, convolutional neural networks are the current state-of-the-art architecture [2]. Due to the fact that there exists a significant amount of research regarding CNNs, known models are comparatively well understood. Known CNNs like ResNets, VGGNets and AlexNets have been academically examined in further detail [45, 108, 67], which enables us to interpret these models with a higher degree of certainty and exactness on another level. It has been shown that CNNs excel in medical image classification. More precisely, ResNets show a convincing performance compared to other CNNs. [90, 44] Next to factors like the data size and the training repetition, the model complexity plays a fundamental role when investigating the training time. Thus one often aims for a rather compact CNN. We note that ResNet with 34 convolutional layers has a performance error of less than 3% better than a ResNet with only 18 layers. Nevertheless, many principles and observations between them are comparable. [44] When training for similarly complex functions the performance improvement by using deeper networks is comparable. The goal of this work is not to reach the highest performance values, but to utilize a model with representable outputs for later experiments. For this reason we implement a ResNet with 18 convolutional layers called ResNet-18 (see fig. 3.1). This way we are able to transfer our won knowledge well to other models and simultaneously perform experiments in a descent time. ResNet-18 expects an input image with the shape HxWx3 (see tab. B.1. W and H are expected to be at least of size 224. Its first convolutional layer (conv1) has a kernel of dimensions 7x7 with a stride of 2. Because it has 64 feature maps it stretches the dimensional ratio to 112x112x64 (neglecting multiple channels for simplicity). Secondly a max pooling operation with a 3x3 kernel (2x2 padding) and a stride of two is used. Batch normalization is also applied. Because of the additional stride in the max pooling operation, the second layer has no additional downsampling and outputs a batch with the dimensions 65x65x128. The following layers consists of two blocks each. Both are characterized by a 3x3 kernel with 2x2 padding on each side. The first block has an increased stride of two, which is why we use a projection shortcuts with a convolutional operation at first and an identity shortcuts everywhere otherwise. After the fifth layer, there is an average pooling operation with a 7x7 kernel and an output size of 1x1x512. We merge all leftover nodes with a 500x1000 fully connected layer followed by a softmax function that outputs the desired number of variables for classification in a one-hot format.

3.2.3 Pipeline

For our experiments in this work, we use ResNet-18 as described above. ResNet-18 has different training and evaluation behavior, such as backpropagation. To switch between these modes, we use the methods model.train() or model.eval(). The model is embedded into a pipeline with several additional features that can be adapted to optimize the pre-processing, training and testing processes. Our pipeline allows for optional pruning that may result in a better performance. Other options are the choice of optimizer and loss function. We weight the loss for “plaque” higher in order to counter class imbalance. Optionally, deterministic computations make the model computations reproducable. The most important feature of our pipeline is the ability to choose which DA techniques are applied in which order. Note that randomization with DA and the use of different hardware components may break determinism. L2 regularization can be used by setting the weight-decay parameter to a non zero value when calling the optimizer. We are also able to vary certain other parameters. Among them are the learning rate, early stopping patience as well as early stopping ACC. Our model performs average pooling in case the images are not of the size 224x224.

Note that our model repeats training with the same data several times, while step-wise adapting the learning rate (ideally scaling with a downward trend). Each repetition is referred to as an epoch. Weight updates are calculated from backpropagation errors and performed batchwise by computing their average for better generalization. Each batch contains n images, where n is commonly a power of two in [32, 64, 128, 256]. For efficiency reasons the data loader preloads the images to the main memory. It is important to note that smaller batch sizes are good against overfitting, but train slower. Larger batch sizes do not always converge as fast, but result in faster training progress. After each epoch, the model is tested on a validation set. The new model is stored in a checkpoint, also including other configurations like scaler and optimizer. The performance is then compared to the model from the same or an earlier epoch that performed best. If the performance improvement lies under a certain threshold early stopping may be performed. If no early stopping is performed a specified value defines the maximum total number of epochs. To generate representative and generalizable results in our experiments, we conduct cross-validation with k subsets and an extra hold-out for testing.

The pipeline tests the trained model automatically after the training process is completed. Performance measurements are logged in a dedicated text file while additionally uploading all metric values to an online service called Weights and Biases (WandB). This enables us to visualize the experimental results in a way that is more easily interpretable. The use of WandB is optional but is of importance, e.g. to detect oerfitting with B-logging early. Here refer to B-logging as the ability to compare the performance of a model when tested on validation as well as the testing set while training.

3.3 Data Preprocessing

In this work we strictly differentiate between pre-processing transformations and those, that actually augment the final data set on which models are trained on. In contrast to DA techniques, Pre-processing transformations are uniformly applied to all images by a possibility of 100%. They substitute the old, unprocessed data and therefore do not augment the data set. The implemented pipeline applies DA techniques by a certain probability, that may be between 0% and 100%. Thus whenever the data loader accesses an image, it may get back different results. Pre-processing highly depends on the data that we apply our model on. Thus we present the underlying data set first.

3.3.1 IVOCT Data Set

The data set we used for our experiments has a total of 3951 individual IVOCT images. Each of these images has a resolution of 347 by 683 pixels and consists of 16 bit unsigned intensity values with a maximum value of 65535. This means that each of the 347 A-scans per frame has a depth of 683 pixels assembled to a B-scan in h5 format. Each image was assigned a label, which indicates whether deposits are present in the vessel, or whether they can be detected based on the single cross-sectoinal image slice. None of the images contains a stent as this would distort results, because the underlying model may learn from the presence of the stent instead of learning the desired property, namely the presence of plaque. In general, six different types of plaque can be distinguished, e.g. if its calcified or not. In this work, we restrict ourselves to a binary classification that distinguishes whether the image represents disease-affected or healthy anatomy. The classification was made by three experienced physicians for each frame. In the classification, a one indicates that plaque was detected, while a zero, on the other hand, indicates its absence. To ensure the images have been labeled correctly, the experts were provided partially the same data for manual B-scans classification. Distinguishing values were classified repeatedly to ensure label quality by labeling with the consensus. A comparatively large data set containing a variety of challenging cases could be acquired this way. For investigations on DA it suffices to merge original characterizations to binary labels “plaque” and “no plaque”. It is important to note that we have 2142 positive and 1808 negative labels. With class weights this is a ratio of 0.542% (with plaque) to 0.458% (without plaque), which shows that we have a slightly imbalanced data set. In the following steps we did not balance the data set for training or testing purposes to avoid information loss, which would inevitably lead to poorer prediction performance. Instead we adapt our training and testing implementations. While training, imbalanced data sets can be handled by first, a weighted random sampler in the data loader, and secondly, by weighting parameters in the loss function. In the following test phase, we adapted the metrics used to measure the performance of our models (as described in the section about metrics). The given IVOCT data consist of multiple intravascular serial images obtained by a different pullbacks and patients. Frames coming from a single pullback cannot be considered independently from each other when they are processed.

3.3.2 Cartesian Transformation

In IVOCT the recorded B-scans (sequence of A-scans) are generated in a 2D polar coordinate format visualized as an image. Using Cartesian transformation the scans can additionally be converted into a human interpretable format. There exist two 2D equations representing the relation of the two coordinate systems. For the relational mapping from the cartesian towards polar coordinates see eq. (3.1). This conversion can mathematically be represented by the transformation matrix given in eq. (3.2).

$$\vec{\nu}\begin{pmatrix} r \\ \theta \end{pmatrix} = r\begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.1}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{pmatrix}\begin{pmatrix} r \\ \theta \end{pmatrix} \tag{3.2}$$

Similarly the backward transformation from the polar into the cartesian coordinate system is shown in eq. (3.4) as a result of the Jacobian matrix as a result of its associated mapping above eq. (3.3). By the trigometric inverse function the sign must be determined in a case distinction depending on which quadrant the point (x,y) is located.

$$\vec{\nu}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \sqrt{x^2+y^2} \\ tan^{-1}\left(\frac{y}{x}\right) \end{pmatrix} = \begin{pmatrix} r \\ \theta \end{pmatrix} \tag{3.3}$$

$$\begin{pmatrix} r \\ \theta \end{pmatrix} = \begin{pmatrix} \frac{x}{\sqrt{x^2+y^2}} & \frac{y}{\sqrt{x^2+y^2}} \\ -\frac{y}{x^2+y^2} & \frac{x}{x^2+y^2} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \tag{3.4}$$

The transformation causes a higher information density towards the centric origin of the image and a lower resolution vice versa. The described mapping has been interpolated with the Makima method which leads to information loss towards the origin, preserving a lower information density vice versa. This inhomogeneous image quality coincides with common qualitative IVOCT imaging observations and is amplified due to limitations of physical principles of OCT scanning techniques (see IVOCT section). A full transformation cycle including forward and backward transformation can create artifacts and lead to information loss. Therefore it is recommended to only transform the polar into cartesian coordinates ones if required and avoid backwards transformation.

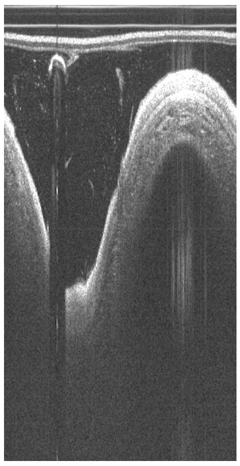


Fig. 3.2: An exemplary IVOCT B-scan (dimensions: 347x683 pixels) in original polar format. Brightness and contrast have histographically been adjusted and distributed more uniform to visualize the B-scan to the human eye.

The resulting cartesian image is scaled during interpolation to a size such that twice the radius corresponds exactly to the smallest side of the polar image. One differentiates between four different types of interpolation, namely: nearest neighbor, bilinear and bicubic. Bicubic interpolation considers the closest 4x4 neighbourhood and builds a weighted average, which forms a good compromise between computational costs and sharpness. After the transformation, the images have an aspect ratio like a equilateral quadrilateral. Thus the entire conversion leaves undefined edges, that must explicitly be defined in preprocessing (see fig. 3.3). A key advantage of this transformation is that the information have a rather realistic relation to each other, leading to a more intuitive appearance. When examining IVOCT image structures, one can distinguish regions of the lumen (signal poor region), media, and plaque in the media (close to the luminal border). Ambiguous information near the detector are weighted less than distant information. This makes the weighting of the information at the origin of the source image more consistent with the actual volume distribution of the anatomy.

3.3.3 Model Preparation

The IVOCT image is a map of intensities and therefore only provides 1D grayscales. Even though ResNet-18 takes an image with resolution 224x224x3 as input size for RGB-images with three channels, we can use it for a 1D image with only a single channel as well. There are two ways to do this. One way is to join the 1D image with two additional zero channels. As described in the section about ResNets, these layers zero out and have no effect on the overall result. Alternatively one can replicate the grayscaled channel, so that all three channels are used for training. This does not add any new information. Depending on the implementation this process may be slower than the training process with the padded image, because additional complex computations have to be executed. When pretraining, replication may result in a better performance, because all information from ImageNet are processed.

We pre-train our network for faster convergence and a higher prediction performance. PyTorch provides an easy way of using a ResNet-18 pre-trained on the ImageNet data

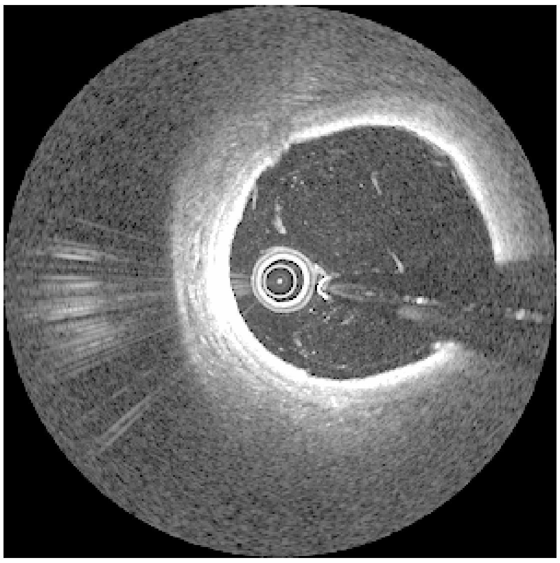


Fig. 3.3: An exemplary IVOCT B-scan (dimensions here: 300x300 pixels) transformed from polar into cartesian coordinate system. The conversion generates undefined edges, here shown in black. Brightness and contrast have histographically been adjusted and distributed more uniform to visualize the B-scan to the human eye.

set. It is unknown if ImageNet data and IVOCT data sets have any intersection of information. Trivially, other IVOCT data sets have more in common with the provided data set, since they stem from a similar domain. It is yet an open question, if pre-training on either of these data sets adds any value to the overall performance.

$$\mathrm{RS} = \frac{D_i - \min}{\max - \min} \tag{3.5}$$

$$\mathrm{MN} = \frac{D_i - \mu}{\max - \min} \tag{3.6}$$

$$\mathrm{SD} = \frac{D_i - \mathrm{mean}}{\sigma} \tag{3.7}$$

When pre-training on similar domains, normalizing the data set with the mean and standard deviation of the pre-training data may as well give better results due to its fine-tuning effect on the model. Re-scaling (RS in eq. (3.5)), Mean Normalization (MN in eq. (3.6)) and Standardization (SD in eq. (3.7)) are good practises to scale input data by standardizing variables. These techniques often reduce training time and the numerical stability of the model [82]. They can prevent increasing generalization error and unstable networks, so that the models prediction probabilities are not biased. As we can see in the above equations, μ corresponds to the mean and σ to standard deviation of the entire data. D_i are the input data, whereas [RS, MN, SD] depict the transformed output data, e.g. image pixels. For this we load the images to a range of [0, 1]. For the ImageNet data set used in PyTorch [74], mean and standard deviation are officially provided in the documentation as mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225]. If not provided, they have to be computed from all images in the data set. Alternatively mean

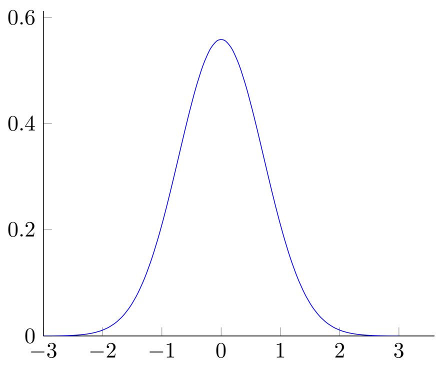


Fig. 3.4: Gaussian curve with mean zero and standard deviation 15/21.

and standard deviation can be set to 0.5. Note that pre-trained models are cached by PyTorch automatically. This implementation allows caching this outside the docker container.

3.4 New Transformations in IVOCT

In the following transformations specifically for IVOCT imaging or data from very similar domains are introduced. Classical DA approaches can be less effective when there is a lack of labeled medical data and fail to generalize easily, because anatomical changes are much more diverse. [8] Hence we aim for new and IVOCT specific DA approaches. We describe their theoretical relation to IVOCT (artefacts, etc.), technical implementation and development.

3.4.1 Tunica Variation

The fundamental idea of this technique is based on the anatomy of the vessel. When considering the shape and diameter of a vessel, it is clear that these characteristics do not directly correlate with whether or not plaque is present in the vessel. CNNs should also perform their classification as independently as possible. Since generalization is increasingly difficult with a smaller number of images, it may be advantageous to be able to randomly change the shape of the vessels. The easiest way to adjust the shape fit of the vessels is to shift the columns of a B-scan up and down in polar form. The difficulty is to choose the sections in such a way that in the end a new and realistic image is created again without distortions being visible or important sections being cut off. We achieve this by three basic steps, which are described in detail below.

The first step is to roughly determine the curve-like form, which can be observed in fig. 3.5. It mainly originates from reflections of the tunica and is extracted by applying several filters. To be able to differentiate it from the bright artefacts caused by the catheter itself, we simply remove the first section of the image. It can be observed that these artefacts remain relatively static in position. By setting the first 75 lines of the image to zero, we simply remove this section. Although we now generate a black bar at the top, this is sufficient to meet the requirements for further processing later on. We apply a Gaussian filter with kernel size 21x21 and a kernel standard variation in X and Y direction of 15 (see fig. 3.4). The Gaussian filter computes the weighted average by in a convolution and thereby blurs the image so that unimportant details are neglected in the following steps. By applying a threshold to the 16 Bit image, we map the image to a binary representation (see fig. 3.5) with the function

$$\mathrm{dst}(P_{src}) = \begin{cases} 0 & \text{if } P_{src} < \text{thresh} \\ \text{maxval} & \text{otherwise} \end{cases} . \tag{3.8}$$

All values smaller than 450 are represented in black (value zero) and values of at least 450 are represented in white (maximum value of 65535). Additionally we identify connected components and determine their statistics with a 4-way connectivity (all pixels that share an edge are considered as neighbours). This is in contrast to an 8-way connectivity, where pixels that share an edge are considered as neighbours as well. Smaller components containing less than 1700 pixels often represent disturbing bright artefacts that are thereby removed.

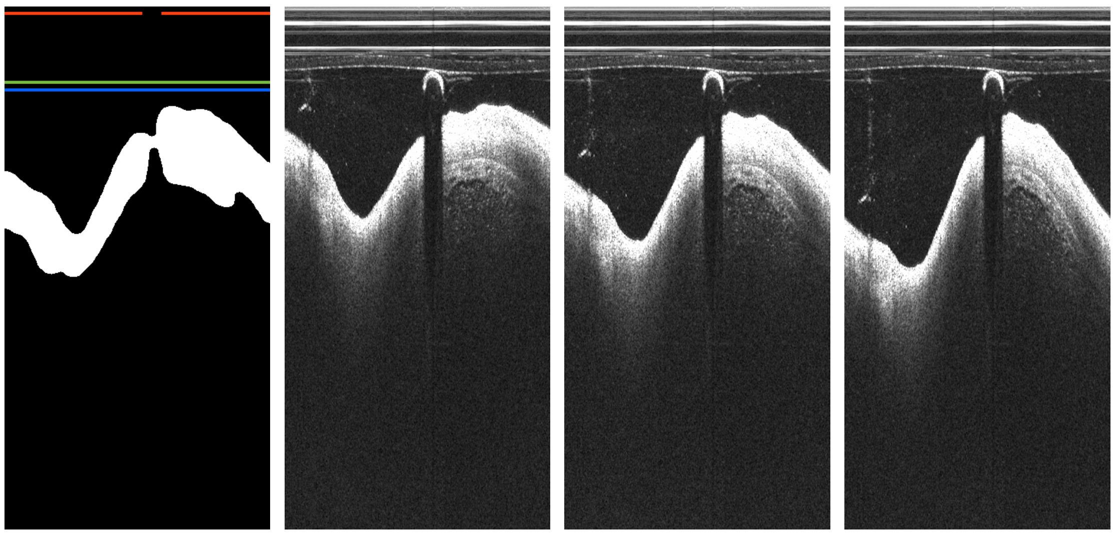


Fig. 3.5: A DA technique varying the form and diameter size of an IVOCT frame in polar format. The first image shows the extracted tunica curve from the original image (third). The original image is compared to the image with size variation showing a positive scaled Hann window (second) and an image with negative scaled Hann window (fourth). Note that all three images on the right are modified for illustration purposes, such that they have a more uniformly distributed histogram.

Before any columns can be shifted, it should be ensured that they can also be authentically extended in the appropriate direction, either upwards or downwards. To avoid unwanted deformation of the image, we exclude two types of columns from the shifting transformation. First we exclude the columns that contain artefacts by the catheters guide wire, because the reflection of the wire itself and its shadow limit how much the columns can be shifted, if the tunica is close to the catheter. Secondly, we exclude those columns where the curve height is above a threshold height of 573 (683 rows in total). This threshold is illustrated by the lowest straight and horizontal line on the most left image in fig. 3.5 (blau). To find the columns that contain the guide wire and the corresponding shadow, we use the original image. The width of the wire is approximately the same in each image, so we assume a static number of 25 columns lined up directly next to each other. It should be mentioned that all column calculations are done using a wraparound, which means that intervals may exceed the boundaries of the image. To determine the position of the guide wire, we set the upper 75 rows of the image to zero again. Then we sum up the gray values for each column and determine the sequence of 25 adjacent columns whose gray value sum results in the smallest overall value. In which sections the curve exceeds the selected threshold can be found out by scanning each column from above. The first white pixel determines the height of the curve in the respective column. If no white pixel is found, the bottom edge of the image is taken as the height. Once we have excluded the columns described above, we look for the largest possible interval (with wraparound) in which the curve does not cross the threshold and no guide wire is contained. This interval is indicated by a red line at the top of the left most image, marking all columns that may be shifted.

Now the columns can be shifted up or down. For an authentic shifted curve we utilize the Hann function, which is a similar approximation to the real curves in the B-scans:

$$h_0(x) = \begin{cases} \frac{1}{L}\cos^2\left(\frac{\pi x}{L}\right), & |x| \leq L/2 \\ 0, & |x| > L/2 \end{cases}. \tag{3.9}$$

The advantage of the Hann function is that it approaches zero and is zero outside the interval $\pm L/2$. Moreover, it has its maximum of one centered in the origin of the x-axis regardless of what value L is equal to. A Hann function with L equal the interval length is moved into the positive direction, such that it starts in the x-y-origin. We take the length of the previously determined column interval and create a series of the same width by inserting the indices of each column into the Hann function. The scalar by which the Hann window is multiplied results randomly from an interval between zero and the maximum distance from the above threshold to the curve. After rounding the values of the Hann window to integer values, the columns are shifted according to the corresponding index either in positive or negative direction.

For shifting a section in a column we set an additional threshold indicated by the second continuous horizontal line (green). In a column, only the pixels below the 683rd row are ever shifted. If the column sections are shifted up by n pixels, we cut them off starting at the threshold. At the bottom, we append pixels accordingly by mirroring the column at the bottom row. When the column is shifted down, the missing pixels are replaced by mirroring the column at the threshold height. Accordingly, the threshold was chosen so that there is always space between the curve and the threshold to take corresponding pixels from there.

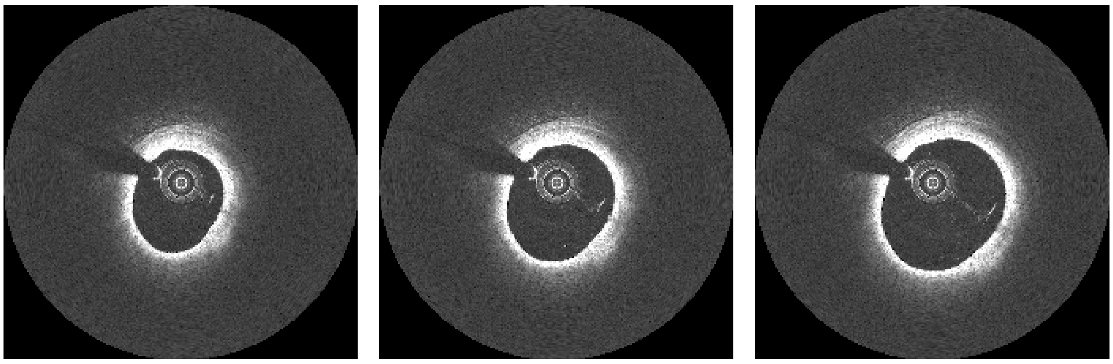


Fig. 3.6: A DA technique varying the form and diameter size of an IVOCT frame in cartesian format. The original image (center) compared to the size variation with a positive scaled Hann window (left) and a negative scaled Hann window (right). Note that all three images on the right are modified for illustration purposes, such that they have a more uniformly distributed histogram.

The final images (see fig. 3.5) can also be transformed into cartesian format as in fig. 3.6, giving a relatively authentic representations of a cross-sectional frame.

3.4.2 Artefact Imitation

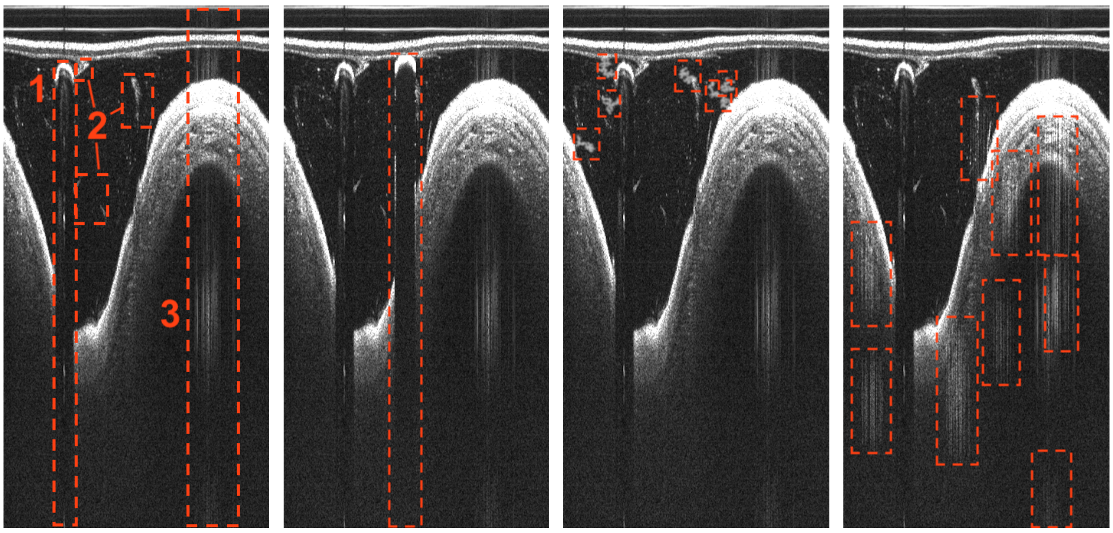


Fig. 3.7: Three DA techniques applied, imitating IVOCT artefacts. The first image shows the original B-scan. The other images illustrate artefact imitations, marked with the red dashed circles. The second frame shows an extra guidewire insertion, the third imitates blood artefacts and the fourth white noisy column areas. Note that all three images on the right are modified for illustration purposes, such that they have a more uniformly distributed histogram.

When having a close look at an IVOCT scan, one can frequently observe three main kinds of artefacts [92]. These can influence the performance of NN models, when training on IVOCT data sets, significantly. One of the most significant artefacts is caused by the guide wire, which leads to a bright reflection in form of a semi circle and a dark shadow, that covers the anatomy behind it. Secondly, one can observe multiple artefacts of various forms that are seemingly random distributed over the entirety of the lumen. Those artefacts are mostly caused by blood residues that reflect light well. Lastly, one can observe brighter groups of columns independently distributed of the actual image content. These brighter fading columns have an overall gradient from the top to the bottom with multiple stationary points. Fading in and out on the horizontal axis is mixed with random deviation. Highlighting is not only additive but depends on the reflection signal itself, so they contain a certain amount of noise.

Based on the described artefacts we introduce multiple ways of augmenting the image data set accordingly. We attempt to recreate the described artefacts by modifying the B-scan in polar format. In this format artefacts are easier to imitate. It further yields the advantage that artefacts that only occur due to the cartesian transformation can be taken into account as well.

Or first approach describes the additional creation of a guidewire reflection and its randomly positioned shadow (see fig. 3.7, frame two). We manually chose an image with a guidewire positioned comparatively close to the catheter and cut it out. This gives us the possibility to insert the cutout randomly into other B-scan in an attempt to make the network invariant to guidewire positioning. To insert the artifact in realistic positions, we make sure to only choose these, where the tunica is far enough away from the catheter. The range of columns where we allow the artifact insertion is therefore computed similarly as described in the tunica variation section with a threshold of of height 613 pixels. Also, we vary the insertion height by 10 pixels to each side.

Next to the guidewire, blood artefacts can be imitated artificially to a certain extend as well (see fig. 3.7, frame three). This is done by creating a Gaussian array (1024x1024 pixels) and masking it with its mean threshold. We then label objects with their size and remove those with a size greater or equal of 1000 pixels. We then convert it with a binary threshold, resize the mask to 319x347 pixels and apply a Gaussian filter with a standard deviation value of one. Afterwards the mask is scaled by a factor of 0.02 and added to the upper half of the original image, which is near to where the blood artifacts usually occur as well. Finally we obtain an augmented image with three to six additional artifacts.

The last artefact we imitate is formed by column groups with irregularly high intensities in different regions (see fig. 3.7, frame four). We imitate this artefact, first, by creating a zero vector for each dimension that has cardinalities equal to the number of columns and rows. Both vectors are supplemented with a Hann window. The first has a width in the range of 40 to 60 is added in a random position and the second has a width between 180 and 220. After both summations, only the first vector is multiplied by an array of the according cardinality that is populated with random samples from a uniform distribution over [0, 1). Then the second vector with a cardinality equal to the number of rows, is replicated to a matrix of the image shape. We perform a multiplication with the first vector and scale it by a factor of five. Finally, the resulting matrix is multiplied with the original image. Previously applied steps are repeated ten times such that we get a set of artifacts we can add to the image.

3.4.3 Histogram Equalization

Histogram equalization may increase the performance of task like image analysis, object detection, and image segmentation [14]. Generally, performance of CNNs can be improved

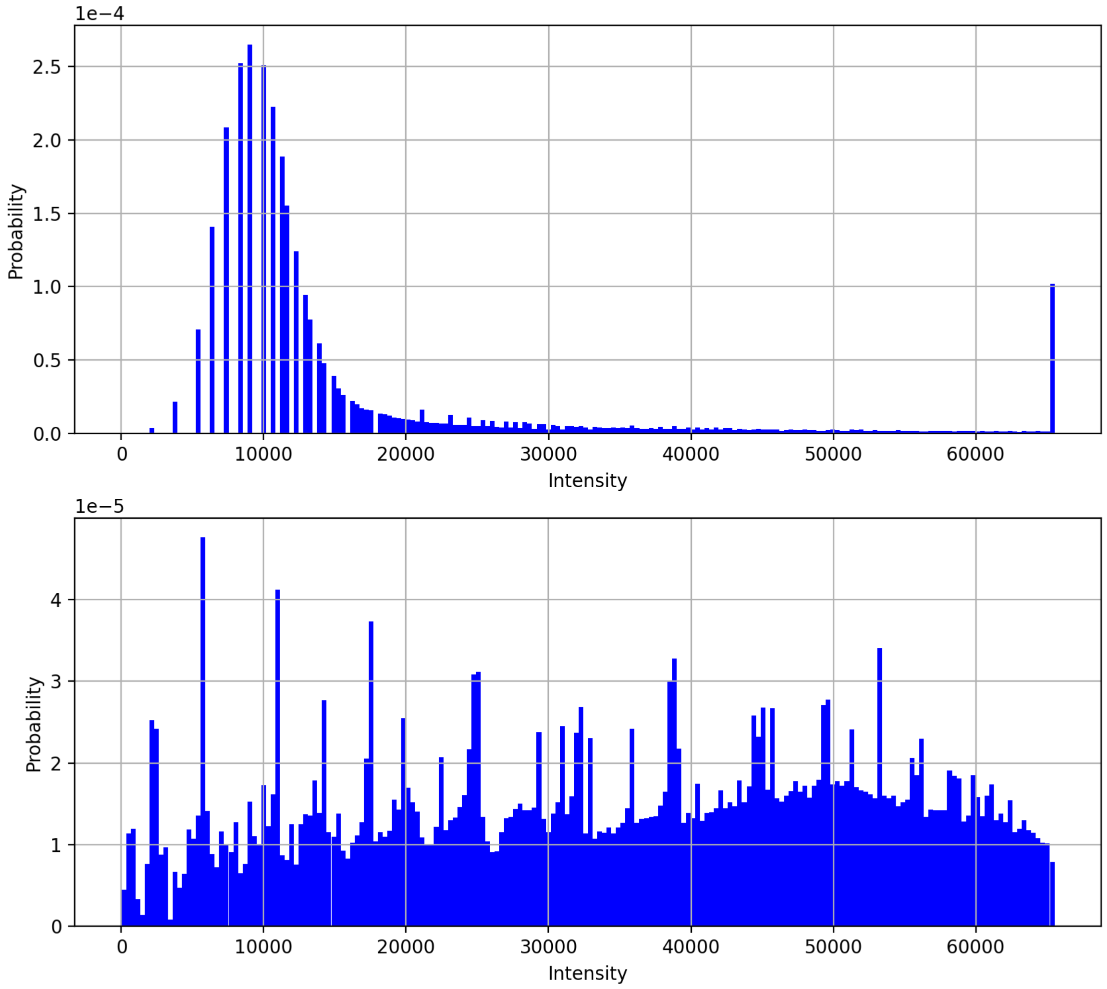


Fig. 3.8: Two IVOCT images histograms with and without a CLAHE filter are compared (upper and lower one).

by pre-processing image data with histogram equalization in scenarios where training data is limited or small [66]. It may help to overcome the limitations of global approaches by performing local contrast enhancement. Histogram equalization is used to improve contrast in images by stretching out local intensity ranges. This increases the contrast of an image, even when it was represented by close values before. Contrast limited adaptive histogram Equalization (CLAHE) redistributes the intensity values of an image by computing histograms corresponding to distinct image sections. It relies on two hyperparameters, the number of tiles and the clip limit. The tile defines the size of image sections considered and the clip limit defines a maximum contrast to prevent noise over amplification.

3.5 Conducted Experiments

Our workflow consists of four main steps, described in the following. First, we subdivide our data to make use of CV with a holdout. Secondly, we test for the mostly suited general model parameters and pre-processing transformations to use for the following experiments. Third, DA techniques are tested and their effects compared. Finally we find an optimal combination of those DA techniques that have a positive impact on the model performance and rule out those with a negative impact.

3.5.1 Subdivision

As our pipeline performs K-fold CV, we have to divide our data set into k mutual exclusive subsets. The expectation is that low values of k would result in a noisy estimate of model performance and large values lead to overly high computational costs. The preferred way would be to freely subdivide the data set and find an optimal value for k, by testing all possible values of k and calculating the covariance. However, this is not applicable due to the data set structure where not all images are completely independent to each other. Thus in our case it makes more sense to decide for a fixed number of $k = 7$ subsets. Since our data set is composed of 56 different unrelated frame sequences we hold out seven of them as a distributed test set. The others 49 sets are used to train the model, where in each CV step seven sequences are chosen as a validation set. Note that all of our DA techniques are online augmented, that is, we only store the original data and execute all transformations in real time as part of the data loader. No actual enlargement of the data set size takes place and thus hard drive storage and resources can be saved. Validation and testset are excluded from DA, but pre-processing transformations are applied to all images.

3.5.2 Hyperparameters

To get decent results, we set a standard initial configuration for our model that is based on generally very common hyperparameters for comparative setups. Cost functions and optimizer are chosen to be cross-entropy and Adam optimizer for all experiments, because they generally performs best on binary image classification CNNs among the adaptive optimizers that automatically tune the learning rate [103]. We prepare our experiments by conducting a variety of tests to find generalized behaviour on hyperparameters (e.g. initial lr or batch size) to narrow down or rule out their influence for simplified analysis.

Some of the unknown model hyperparameters we want to optimize in pre-tests are values for the initial learning rate and batch size, since these have by far the greatest influence on training behavior. Another question to be answered is if pre-training our model with ImageNet gives any benefits. Related to this, we normalize our data sets mean and standard deviation to either 0.5 or ImageNet itself with mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225]. We further check if rescaling, mean normalization or standardization leads to the highest model performance, when conducting supervised learning with IVOCT data. Since we use ResNet-18 for our experiments, it remains unclear whether the model benefits from higher performance if we either pad or if we replicate the image to a 3-channel image. Our PyTorch pipeline uses ACC as a measure for early stopping. We want to find the best parameters for early stopping patience as well as early stopping ACC. The threshold (early stopping ACC) defining how long the model continues training is chosen simply by observing the models variance at the time where the validation ACC for each epoch is around its highest value. To make sure early stopping is triggered early enough, we double the observed variance and take it as stopping ACC. Early stopping patience is tested for a range of values between x and y epochs. We observe around which epoch the models validation accuracy finds its maximum before decreasing and how many epochs it takes such that we can assume that no further improvement takes place. This number of epochs is chosen as early stopping patience. Through this approach it is made sure to recognize when the model still improves slowly with a slight oscillation and not stop too early. The training process stops at a maximum of 50 epochs. After the training has ended, the model with the least validation loss during training was used avoiding excessive overfitting to make predictions on the test set. From the results further performance measures are derived. Because early stopping and L2 regularization have similar effects on generalization, we perform two different types of tests, that is: run a first experiment where we only apply L2 regularization and second, run one only with early stopping. For comparability, we employ dropout just like Gessert did [35] with a probability of p=0.5 before the output of every model.

3.5.3 Tests

Finally it is left open if training with cartesian images is always to be preferred over images in polar format. Customized multi-path architecture using both representations for an effective feature exploitation may be developed further in the future [35]. We therefore consider both variants if possible and test DA techniques tailored to each representation if applicable. It is important to mention that transformation to cartesian format changes the maximum value due to cubic interpolation of the image drastically. Most filters applied to images in cartesian format distort the circles border and when they are applied before the the transformation their effect might be reduced due to interpolation, especially towards the center of the image. A Gaussian convolution filter, for instance, blurs and thus distorts the edge of the circle to the extent of its kernel diagonal in pixel units. This is why the order of applied transformations including DA and pre-processing might effect the models performance. Therefore we note that the order of pre-processing steps plays a crucial role.

But not only the order of pre-processing transformations must be considered. The application sequence of DA techniques has to be taken into account as well. Its importance can be clearly explained by considering a B-scan in polar representation. If it is converted to cartesian format in the way described above, the upper pixel information is squeezed to the center of the image. In the case we flip the image before, exactly the opposite effect happens. The pixels previously positioned at the bottom of the image are squeezed to its center. If, on the other hand, the image is flipped after the polar transformation takes place, this is equivalent to shifting the polar image by half of the horizontal image width beforehand.

After suitable hyperparameters without DA have been determined, detailed DA tests can be performed. In order to be able to meaningfully evaluate the effect of individual techniques, the tested models are compared with a suitable reference model in each case. The reference model can be trained with DA, but is not required to be. The training data of the compared models only differ in the respective investigated method. To obtain meaningful results, some techniques may be combined. By transformations using random variables, the results of some tests vary when run repeatedly. However, these are marginal due to repeated training in multiple epochs and can therefore be neglected. In the following, we list which DA techniques are tested. This is to determine how large their positive or negative effect is on the performance of the generated model. Using the knowledge gained from the behavior of the methods, an optimal strategy of combined DA techniques for training with IVOCT data is developed.

First, simple **affine transformations** are tested. This includes shifting and rotation in different intensities. Depending on the results of the first tested affine transformations, we examine further IVOCT specific affine transformations, such as stretching, shearing and scaling, again in different intensities. They take advantage of certain properties and commonalities of the individual frames. Firstly, less or no information about the anatomy can be found in the lower part of the frames. It denotes the vertical multiplication of all y-coordinates, where the upper edge of the frame represents the x-axis with $y = 0$. After interpolation an overlap by n pixels is cut off at its lower edge. n is randomly chosen from a given interval. B-scans have similar feature positions along the temporal dimension that fit well within the context of the other A-scans. Thus, horizontal shearing scales the image from a centered origin along the temporal axis cutting off a random number of $x \in [0, n/2]$ overlapping columns after bilinear interpolation. The new images are similar to the original one, but comes with varied waveforms. Because left and right edges of the frame do no longer fit together after applying the cartesian transformation, further artifacts emerge. Scaling differs from the other transformations in that it crops all image edges instead of specific sides only. Secondly, we augment our data set by applying the **elastic deformation** transformation by Simard [84]. Different types of non-linear **intensity shifts**, such as contrast jittering, brightness jittering, CLAHE and Davella's filter [25] are investigated. Furthermore we test whether **convolutional filters** such as Gaussian noise, salt and pepper noise, and gaussian blur have any measurable effects on the performance. Next, the effect of **partial masking** (random and guide wire removal) is investigated. Finally, new techniques are applied with additional complexity. These include **artefact imitation** like tunica variation, intense columns and blood speckle.

4 Results

In the following the successful application of DA techniques is demonstrated. Hyperparameters and generalization capabilities of the trained CNNs architectures are are tested and most promising values are determined. Based on the augmentation experiments the performance of different techniques is tested. From these results, a strategy is determined and its validity proven.

4.1 Underlying Model

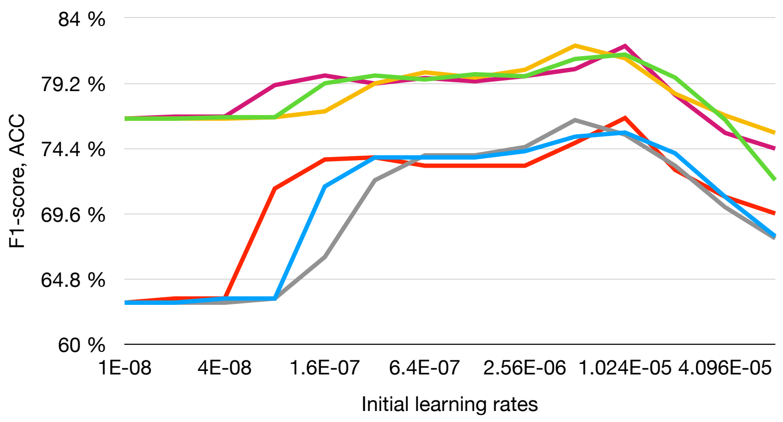


Fig. 4.1: Plot of F1-score and ACC for grid search of initial learning rate against batch size. Each line indicates a different batch size. Purple, green and yellow indicate F1-score, while red blue and gray represent the ACC for a batch sizes of 64, 128 and 256.

Important hyperparameters are determined by restricted grid search (see fig. 4.1) with a coarse grid using the F1-score from simple training with validation set and an extra holdout to keep computational costs within the acceptable range. These hyperparameters are the initial learning rate and batch size. We test batch sizes between 64 and 256 in powers of two. The lower limit is set to prevent the loss from fluctuating excessively and the upper limits ensures rarely occurring features can be taken into account as well. Initial learning rates are chosen between 0.5e-8 and 1e-5 with a step size as power of two. This results in 14 different initial learning rates for each batch size. Performance measures of 42 different models that have been trained in total are plotted in a graph (see fig. 4.1). Based on the generated generated curve, the model that performs best on the combination of both, initial learning rate and batch size, is picked as starting point for training. Transfer learning is used with ImageNet and a standardizes IVOCT data set in cartesian representation. No dropout or L2 regularization have been considered. The F1-score and ACC are plotted against the initial learning rate for each model in fig. 4.2.

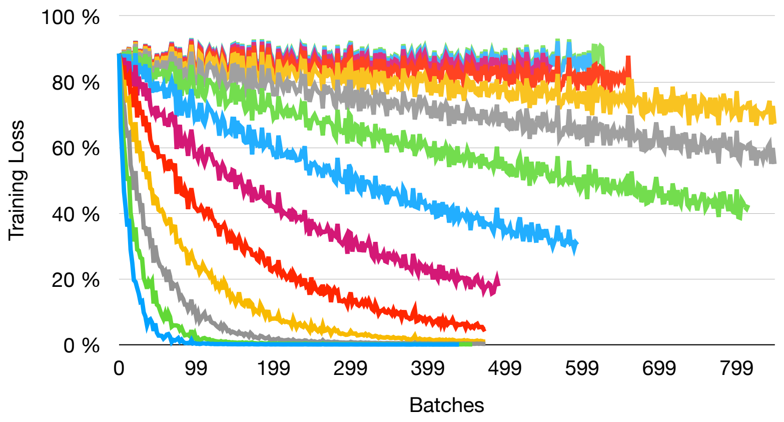


Fig. 4.2: Losses calculated while training with distinct initial learning rates on a batch size of 128 samples. They decrease faster if the initial learning rate is higher and vice versa.

We observe a similar curve for all three batch sizes. An initial learning rates smaller than 3.2e-7 comes along with a decreasing performance for F1-score as well as ACC. Larger learning rates lead to peak results before they drastically decrease the performance. It can be seen that a larger batch size tendentiously leads to increased overall performance for particularly small batch sizes, but does not differ much if initial learning rates were increased.

The initial learning rate of 3e-6 and a batch size of 128 samples is chosen from the best performing batch size in the center of the curve. This results in 19 distinct batches for each epoch. The red line represents a well formed and slowly converging curve and belongs to the chosen initial learning rate.

Early stopping has a similar effect as L2 regularization, when choosing the model with the best validation performance as final test model. Due to this reason we take the optimized hyperparameters from above and test training the ResNet with and without L2 regularization to see if it gives us any additional performance improvement. For the weight decay we choose values from 1e-4 and increase by factors of 10 until 1e-1 is reached. Our tests have shown that models perform best with only slight weight decay levels of 1e-2, returning the best MCC score with slight improvement of 0.5%.

Tab. 4.1: Averaged performance scores of several model initialization and pre-training variants. Pre-training variants are rescaling (resc), mean normalization (norm) or standardization (std). The mean values for standardization are either 0.0, 0.5, or congruent to those of ImageNet (IN). Padding (pad) and transfer learning (trans) may likewise result in different outcomes. Bold numbers indicate the best performance value for the respective metric.

Distribution	SENS	SPEC	PPV	BACC	F1-score	MCC
resc	0.5566	0.7166	0.6598	0.6366	0.5959	0.2902
resc pad	0.6139	0.6210	0.6634	0.6175	0.6010	0.2610
resc trans	0.6632	0.6577	0.7045	0.6604	0.6589	0.3356
resc trans pad	0.5270	**0.7719**	0.6956	0.6495	0.5561	0.3109
norm	0.6211	0.7387	0.6916	**0.6799**	0.6578	0.3630
norm pad	0.5976	0.7156	0.7024	0.6566	0.6191	0.3559
std 0	0.5802	0.7167	0.6880	0.6484	0.6179	0.3145
std 0 pad	0.6800	0.6752	0.7095	0.6776	**0.6792**	**0.3637**
std 0.5	0.6920	0.5991	0.7202	0.6455	0.6393	0.3103
std 0.5 pad	**0.7058**	0.6318	0.7362	0.6688	0.6583	0.3451
std 0.5 trans	0.6858	0.6176	0.6859	0.6517	0.6691	0.3171
std 0.5 trans pad	0.5817	0.6714	0.7067	0.6266	0.5585	0.2704
std IN trans	0.6938	0.6245	0.6990	0.6592	0.6812	0.3353

The same environment has been used to test if rescaling, mean normalization or standardization lead to the highest performance (see tab. 4.1). Rescaling refers to transforming the intensities to a range between zero and one, while normalization sets the mean to zero and scales the image to be between minus one and one. Alternatively we may standardize our image to have a standard deviation of one $\sigma = 1$ and mean $\mu = 0$. We can use the same mean and standard deviation of ImageNet with $\sigma = [0.229, 0.224, 0.225]$ and $\mu = [0.485, 0.456, 0.406]$ when standardization is applied or simply standardize our data with $\sigma = 0.25$ and $\mu = 0.5$. If we standardize with ImageNets hyperparameters, transfer learning by initializing pre-trained weights and biases may be advantageous. Otherwise they can be initialized randomly. This way it is examined if any learned ImageNet features can be transferred when training on IVOCT data. Note that transfer learning for mean normalized images is not considered since then the initial activations distribution seriously deviates from ImageNet. Furthermore 2D images may be replicated or padded to three channel tensors. Both variants are tested because it is unknown whether the model can benefit from either of them. Rescaling and transfer learning are used in the following experiments. This results in the average metrics: PPV 0.6425%, SENS 0.6158%v, SPEC 0.6144%, F1-score 0.6236%, BACC 0.6151% and MCC 0.2344%. Considering the validation accuracy we observe, that an early stopping patience of 30 epochs suffices and no improvement can be expected from continued training.

4.2 Generalization with Validation

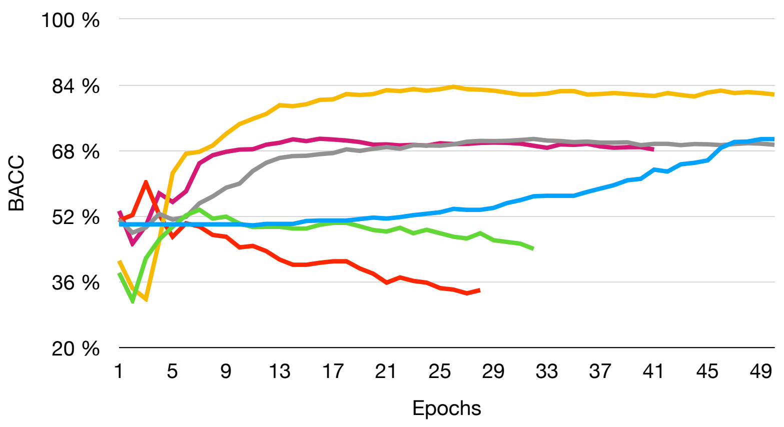


Fig. 4.3: Validation performance (BACC) of training on a data set using CV plotted against number of epochs.

CV was performed and tested for each fold after each epoch on the validation set. The resulting BACC values were plotted against the epochs. A fold is plotted in a separate color and the epoch number starts at one in each case. It can be seen that the BACC usually fluctuates heavily at the beginning, but then quickly levels off and steadily drifts up or down. It can be seen that all folds have a maximum above 0.5 BACC. However, the BACC then commonly drops again at different rates or settles in a certain range.

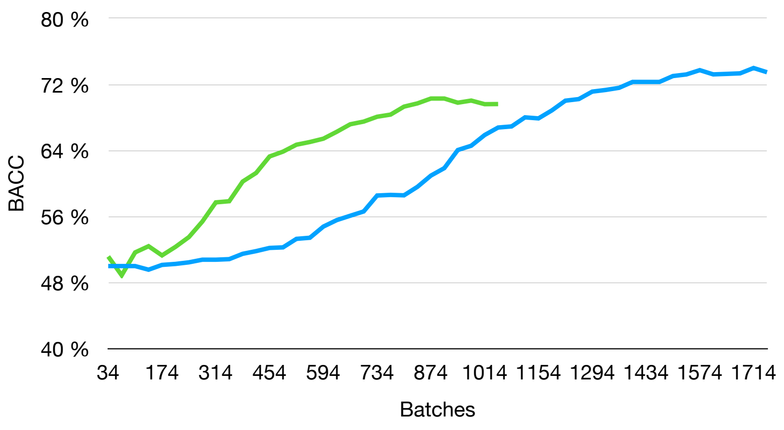


Fig. 4.4: Validation performance of training on a data set with and without DA (blue and green) plotted against number of batches.

ResNet is trained on the same data set with same hyperparameters two times (see fig. 4.4). The two curves describe the BACC validation performance for one test conducted with and one test conducted without DA. Fully random horizontal shifting is applied in the second test (blue). A single holdout is used for validation after training each batch. We observe that the validation performance increases faster when training without DA. On the other hand when applying DA the validation performance increases slower but is able to reach a higher maximum. Tests confirmed that a lack of DA result in early overfitting and poor generalizability (green curve). By using data augmentation, this effect could also be reduced when training with IVOCT data. However some show the opposite effect. Thus is cannot be generalized for all DA techniques.

4.3 Augmentation Experiments

As reference values, we use the performance values of the models trained with augmented data among each other and thus compare their effectiveness in a more meaningful way. In the following, each tested data set was augmented by a different transformation and their performance compared using CV and standard metrics as described above. We apply several transformations to our B-scans before transforming them into cartesian format. By this it is ensure that we do not alter the form of zeroed edges in cartesian format, which would significantly differ from realistic images and is assumed not to have any performance improving impact on the model.

The confusion matrix is used to compute SENS as well as SPEC to compare how well true positives and true negatives can be identified. Precision (PPV) measures the proportions of positive and negative results overall. Furthermore we use BACC instead of ACC which considers both SENS and ACC and joins them in one measure independent of weight, because our data set is slightly unbalanced. Additionally we utilize the F1-score measure returning high values if the number of true positives obtained is high compared to the other confusion matrix cells. MCC represents a recently introduced measurement, which covers F1-score and BACC as well. For this reason we consider it in the following as well. Since it deteriorates with unbalanced data sets, it has to be considered carefully between different data sets. [18]

4.3.1 Affine Transformations

Horizontal Shifting

Tab. 4.2: Performance measures of ResNet trained on data sets that have been augmented by random horizontal shift value from distinct pixel ranges. Training with both, polar (Pol) and cartesian (Car) representation has been tested. Bold numbers indicate the best performance value in the respective column.

Repr.	Pixels	SENS	SPEC	PPV	BACC	F1-score	MCC
Pol	0	0.7891	0.4567	0.7422	0.6229	0.7649	0.2549
	5	0.7846	0.4567	0.7411	0.6207	0.7623	0.2495
	20	0.7786	0.4507	0.7375	0.6147	0.7575	0.2367
	173	**0.8087**	**0.4925**	**0.7595**	**0.6506**	**0.7834**	**0.3127**
Car	0	0.9397	0.2537	0.7140	0.5967	0.8114	0.2760
	5	0.9398	0.2537	0.7140	0.5967	0.8114	0.2761
	20	0.9503	0.2478	0.7146	0.5990	0.8158	0.2919
	173	**0.9428**	**0.2955**	**0.7262**	**0.6191**	**0.8204**	**0.3270**

As horizontal shifting corresponds to centre rotation in the cartesian representation. We perform shifting operations on polar representations to avoid information loss due to repeated interpolation. We determine the shifting amount and direction (horizontally or vertically) by choosing a random value from chosen range. For instance, when a range of 20 pixels is defined, the images are shifted on the horizontal axis in positive or negative direction by a random value in the interval -20 and 20. As Davella [25] proposes to give increased weight to smaller shifts, we test if shifting the images by a smaller amount gives the same results in unproportional improvements compared to the pixel range size. This is done by testing for different ranges 5, 20 and 173, where the last one is the maximum possible amount.

Slight rotations (5-20 pixels) result in only minor to no changes in balanced prediction accuracy. When training with cartesian representations, MCC increases by 5.1%, while BACC improves by 2.2%. Training on polar representation behaves similar and results in a fundamentally larger SPEC, but smaller SENS. We observe that the model can learn best from fully random cartesian rotations with maximum displacement of 173 pixels in positive as well as negative direction.

Vertical Shifting

Tab. 4.3: Performance measures of ResNet trained on data sets that have been augmented by random vertical shift value from distinct pixel ranges. Training with both, polar (Pol) and cartesian (Car) representation has been tested. Bold numbers indicate the best performance value in the respective column.

Repr.	Pixels	SENS	SPEC	PPV	BACC	F1-score	MCC
Pol	0	0.7891	0.4567	0.7422	0.6229	0.7649	0.2549
	5	0.7725	0.4597	0.7391	0.6161	0.7555	0.2381
	20	0.7740	0.4567	0.7385	0.6154	0.7558	0.2370
	341	**0.8448**	**0.4208**	**0.7430**	**0.6328**	**0.7906**	**0.2920**
Car	0	0.8192	0.4059	0.7321	0.6126	0.7732	0.2435
	5	0.8539	0.3791	0.7316	0.6165	0.7880	0.2637
	20	0.8614	0.3850	0.7352	0.6232	0.7933	0.2804
	341	**0.8765**	**0.3820**	**0.7376**	**0.6292**	**0.8011**	**0.2996**

Additionally shifting can be applied vertically. The augmentation technique is applied on the image in polar representation. Again, the model is trained on both representations, one with and one without transformation to a cartesian representation. Shifting intensities are scaled similarly to the horizontal shifting by 5,20 and 341 pixels. However, it must be taken into account that the resolution before the cartesian transformation differs in height and width. Therefore, a different maximum must be selected as the full shifting rate.

Slight vertical shifts of up to 20 pixels in any direction result in only minor negative to no changes in balanced prediction accuracy. The MCC shows similar behaviour while achieving a maximum of around 3.5% improvement. When training with cartesian representations, larger improvements can be achieved. MCC increases by 5.6%, while BACC improves by 1.6%. We observe that the model can learn best from fully random vertical shifts with maximum displacement.

Overall, it should be noted that horizontal and vertical shifts have similarly strong positive effects on performance (see tab. 4.3). However, this only applies to full shifts. Smaller shifts might even have a negative effect when applied for certain test sets. A negative offset of the measurement series performance in SPEC and MCC with vertical shifts is caused by differences in the cartesian transformation and resulting differently positioned images. No measurable measurable improvement can be observed, when training on IVOCT data with a combination of horizontal and vertical shifts.

Random Flipping

Tab. 4.4: Performance indicators of ResNet trained on data sets that have been augmented by fully random horizontal shifts combined with vertical and horizontal random flipping. Training with both, polar (Pol) and cartesian (Car) representation has been tested. Bold numbers indicate the best performance value in the respective column.

Repr.	Axis	SENS	SPEC	PPV	BACC	F1-score	MCC
Pol	Shift only	0.6878	0.5940	0.7759	0.6409	0.7211	0.2813
	Horizontally	**0.7042**	**0.6026**	**0.7847**	**0.6534**	**0.7342**	**0.3071**
	Vertically	0.6913	0.5997	0.7812	0.6423	0.7235	0.2924
Car	Shift only	0.7212	0.6072	0.7901	0.6642	0.7473	0.3281
	Horizontally	**0.7250**	**0.6047**	**0.7899**	**0.6649**	**0.7496**	**0.3294**
	Vertically	0.6201	0.6495	0.7922	0.6348	0.6701	0.2805

For our next experiments, a data set augmented by fully random horizontal shifting was modified by additional horizontal and vertical flipping. We again test the training with and without subsequent transformation into cartesian format. In tab. 4.4, flipping is omitted in the first experiment. This allows us to make meaningful comparisons and evaluations of the effects of flipping.

Models trained with horizontally flipped images perform best in comparison. They improve the performance in both cases, 2.5% trained on polar representation and 0.2% on cartesian representation. With vertical shifting, however, a positive effect can only be observed when training in polar representation. When vertically flipped images are transformed into cartesian format, the sensitivity and the MCC score deteriorate drastically.

Stretching, Shearing, Scaling

Tab. 4.5: Performance indicators of tested models trained on data sets augmented by downward (Dw) stretching, horizontal (hor) shearing and scaling. Transformations were applied to polar representation. The numbers 25, 30, 70, 100 and 200 represent the maximum number of rows or columns (n) that were cut off in total. Training on polar (Pol) and cartesian (Car) representation is considered.

Repr.	DA technique	SENS	SPEC	PPV	BACC	F1-score	MCC
Pol	no DA	0.6870	**0.5838**	**0.7737**	0.6354	0.7171	0.2697
	Stretch Dw 25	0.6949	0.5757	0.7715	0.6353	0.7212	0.2698
	Stretch Dw 70	0.7016	0.5717	0.7720	0.6367	0.7248	0.2739
	Stretch Dw 200	**0.7145**	0.5642	0.7729	**0.6393**	**0.7315**	**0.2818**
	Shear Hor 10	**0.6850**	0.5829	0.7725	0.6340	0.7157	0.2668
	Shear Hor 30	0.6842	0.5881	0.7745	0.6361	0.7165	0.2707
	Shear Hor 100	0.6831	**0.5945**	**0.7769**	**0.6388**	**0.7176**	**0.2757**
	Scale	0.6856	0.5875	0.7748	0.6366	0.7173	0.2718
Car	no DA	0.7068	0.5480	0.7589	0.6274	0.7266	0.2544
	Stretch Dw 25	0.7073	0.5504	0.7601	0.6288	0.7272	0.2576
	Stretch Dw 70	0.7086	0.5561	0.7629	0.6324	0.7286	0.2653
	Stretch Dw 200	**0.7111**	**0.5672**	**0.7684**	**0.6391**	**0.7313**	**0.2800**
	Shear Hor 10	**0.7050**	**0.5481**	**0.7586**	**0.6265**	**0.7249**	**0.2531**
	Shear Hor 30	0.7016	0.5482	0.7580	0.6249	0.7214	0.2507
	Shear Hor 100	0.6895	0.5488	0.7560	0.6192	0.7094	0.2421
	Scale	0.7080	0.5537	0.7617	0.6309	0.7280	0.2621

With stretching and shearing we test further IVOCT specific affine transformations. Different interval sizes should help to better estimate how strong the stretching must be such that the model can learn from it. The test results show that augmentation by stretching and shearing can have a positive effect on the performance of the model when training polar images. Thus, stretching increases the MCC by 1.2%, while shearing increases the MCC by 0.9% when trained with images in polar transformation (see tab. 4.5). Stretching also increases the performance when training with data in cartesian representation. Shearing on the other hand shows a negative effect. BACC and F1-score reflect the described tendencies similar to MCC. However, if SENS, SPEC and PPV are considered separately, the described tendencies do not apply.

4.3.2 Elastic Deformation

Similar to affine transformations, elastic transformations attempt to increase the learning effect for the model by moving and reshaping features. A mesh grid was created from two Gaussian filters, from which the shifted coordinates for the new pixel positioning emerge. By subsequent interpolation we generate images congruent to Simards proposal [84]. The deformation is conducted before the cartesian transformation is applied. We tested this transformation with an alpha values around 500 and a sigma value around 25. Several experiments were performed using different alpha and sigma values. Both values were varied by increments of ten percent from minus 60% to plus 140% (rounded). The elastic deformation transformation increased the performance of the MCC, BACC or F1-score in all of the cases. It had a small positive effect on the test result. A test with initial values scaled by 30% for alpha and sigma results in peak improvement of 2% MCC to an average of 0.2697. Training on data in cartesian representation does results in an improvement smaller than with polar transformation. Its peak performance is reduced by 0.93%.

4.3.3 Intensity Shifting

Tab. 4.6: Performance indicators of tested models trained on data sets augmented by different intensity shifts. Transformations were applied to polar representation. Training on polar (Pol) and cartesian (Car) representation is considered. The Lowest values have been

Repr.	DA technique	SENS	SPEC	PPV	BACC	F1-score	MCC
Pol	no DA	0.6870	0.5838	0.7737	0.6354	0.7171	0.2697
	Contrast 0.1	0.6856	0.5842	0.7733	0.6349	0.7161	0.2686
	Contrast 0.3	**0.6829**	0.5851	0.7724	0.6340	0.7143	0.2665
	Contrast 1.0	0.8825	**0.2388**	**0.6968**	**0.5607**	**0.7787**	**0.1570**
	Brightness 0.1	0.6846	0.5824	0.7725	0.6335	0.7147	0.2662
	Brightness 0.3	0.6800	0.5796	0.7702	0.6298	0.7099	0.2592
	Brightness 1.0	**0.6637**	**0.5697**	**0.7619**	**0.6167**	**0.6932**	**0.2347**
	CLAHE	0.5914	0.6656	0.8001	0.6385	0.6452	0.2795
Car	no DA	0.7068	0.5480	0.7589	0.6274	0.7266	0.2544
	Contrast 0.1	0.7044	0.5494	0.7575	0.6279	0.7266	0.2543
	Contrast 0.3	0.7037	0.5483	0.7566	0.6270	0.7248	0.2522
	Contrast 1.0	**0.7023**	**0.4040**	**0.6810**	**0.5537**	**0.7872**	**0.1427**
	Brightness 0.1	0.7044	0.5466	0.7577	0.6255	0.7242	0.2509
	Brightness 0.3	0.6998	0.5438	0.7554	0.6218	0.7194	0.2439
	Brightness 1.0	**0.6845**	**0.5329**	**0.7481**	**0.6077**	**0.7037**	**0.2204**
	CLAHE	0.6122	0.6288	0.7843	0.6305	0.6537	0.2652

We investigate a variety of variants to augment the data set by intensity shifting. In our experiments, contrast and brightness jitering is used in the attempt to make the model robust to any of those variations. We test whether and if so, which of the intensity shiftings help the model to learn better. The floating point numbers define a maximum value of the interval from which a jittering intensity is chosen. Furthermore it is investigates weather intensity shiftings introduced by Davella and CLAHE result in an improved performance or not. CLAHE is an exception in this context, since transformed and realistic images are too distinct and CLAHE images therefore unsuitable for an extension of the data set. Nevertheless, it may be that the model benefits from this pre-procesing transformation, since it highlights particularly low-contrast sections clearly.

Tab. 4.6 shows that neither contrast nor brightness jittering has a positive effect on the performance of the model. If the interval from which the jittering value is selected is small, this has only a very small effect on the MCC values when training with polar as well as when training with Cartesian representation. Jittering from an interval of increased intensities from 0.3 to 1.0 lead to a drastic reduction of BACC and MCC. The most drastically reduced numbers are marked by bold numbers. Particularly strong contrast has the effect of reducing the BACC by 9% and the MCC by 11%. The CLAHE filter clearly improves increased BACC and MCC values. All regularities are similarly observable in polar as well as Cartesian representation.

4.3.4 Noise Filters

Tab. 4.7: Performance indicators of tested models trained on data sets augmented by Gaussian blur (B), noise (N) as well as Salt and Pepper noise. Training on polar (Pol) and cartesian (Car) representation is considered. Transformations were applied to the according representation.

Repr.	DA Technique	SENS	SPEC	PPV	BACC	F1-score	MCC
Pol	no DA	0.6870	0.5838	0.7737	0.6354	0.7171	0.2697
	Gauss N 1000	0.5478	0.6964	0.7643	0.5871	0.5483	0.2382
	Gauss N 2000	0.5360	0.7087	0.7710	0.5874	0.5361	0.2404
	Gauss B S1	0.7177	0.5586	0.7618	0.6282	0.7308	0.2580
	Gauss B S2	0.7916	0.4126	0.7025	0.5921	0.7994	0.2495
	Davella	**0.6860**	**0.5787**	**0.7722**	**0.6332**	**0.7139**	**0.2655**
	Salt and Pepper	**0.6721**	**0.5795**	**0.7701**	**0.6381**	**0.7206**	**0.2652**
Car	no DA	0.7068	0.5480	0.7589	0.6274	0.7266	0.2544
	Gauss N 1000	0.5496	0.6636	0.7520	0.5815	0.5598	0.2354
	Gauss N 2000	0.5778	0.6759	0.7587	0.5818	0.5476	0.2376
	Gauss B S1	0.7395	0.5058	0.7495	0.6226	0.7423	0.2452
	Gauss B S2	0.8934	0.3598	0.6902	0.5865	0.8109	0.2367
	Davella	**0.7058**	**0.5429**	**0.7574**	**0.6252**	**0.7234**	**0.2502**
	Salt and Pepper	**0.6939**	**0.5467**	**0.7578**	**0.6325**	**0.7321**	**0.2524**

Multiple noise filters are applied to test if the ResNet benefits from this kind of augmentation. Gaussian blur (B) is applied with kernel size 13 and different standard deviations $\sigma = 1$ and $\sigma = 2$ such that it can be investigated if the benefit is either in smaller or larger blur intensities. For Gaussian noise (N) multiplicative factors 1000 and 2000 are chosen. Furthermore the effect of Davellas noise was tested as alternative to Gaussian noise as well as salt and pepper noise. Gaussian blur and Davellas [25] filter were applied before the cartesian transformation. Salt and pepper noise was applied to the corresponding representation on which the model was trained on to avoid removal by interpolation.

Our test results were not improved by training with Gaussian Noise, Blur filter. Instead, they made the model learn slightly slower. However, this changed little in the model performance. Davellas noise as well as salt and pepper noise show no or only negligible added value during training (bold numbers).

4.3.5 Partial Masking

Tab. 4.8: Performance indicators of tested models trained on data sets augmented by partial masking. Training on polar (Pol) and cartesian (Car) representation is considered. Transformations were applied to the according representation.

Repr.	DA Technique	SENS	SPEC	PPV	BACC	F1-score	MCC
Pol	no DA	0.6870	0.5838	0.7737	0.6354	0.7171	0.2697
	Random S0.2	0.6930	0.5723	0.7703	0.6326	0.7184	0.2654
	Random S0.4	0.6969	0.5510	0.7626	**0.6239**	0.7163	**0.2498**
	Remove Guidewire	0.4312	0.7433	0.7824	0.5872	0.5332	0.1791
Car	no DA	0.7068	0.5480	0.7589	0.6274	0.7266	0.2544
	Random S0.2	0.7148	0.5395	0.7580	0.6256	0.7294	0.2533
	Random S0.4	0.7187	0.5181	0.7503	**0.6169**	0.7273	**0.2377**
	Remove Guidewire	0.5065	0.6563	0.7469	0.5814	0.5796	0.1636

As with the other experiments, we train a model with data modified to a 50 percent probability by partial masking. In partial masking, different masking sizes are tested to avoid removing too much relevant information from the training data. The size of the coverage window is randomly chosen similar to Davella [25]. Each side has a length corresponding to a maximum of n and a minimum of $n/2$ pixels, where n is a percentage of the respective edge length in pixels. First, n is chosen as 0.2% (S0.2) and then as 0.4% (S0.4). In addition, we test whether removing the guidewire that was applied by Davella [25] as well has any advantages. In our experiments, the null masking covers 30 contiguous columns, assuming that the last column is contiguous with the first. The algorithm that determines which location is covered was implemented in the same way as described in the section about tunica variation.

When validation performance is observed, it can be seen that the model learns significantly slower when training with partial masked data than when no partial masking is applied. This is also reflected in the results, especially BACC and MCC, in tab. 4.8. Neither in polar nor in cartesian representation training with partial masking augmented data leads to an improvement in performance. Small masking windows change little or nothing in the results. Larger maskings reduce the learning speed, but also do not improve the test results. Thus, when the same epoch is used to run tests, significantly lower MCC values are achieved (bold numbers). Random guidewire removal has an even worse effect on test results in our experiments, reducing BACC and MCC by more than 5%. Guidewire removal has similar effects as partial masking and does not lead to any detectable performance boost.

4.3.6 Artefact Imitation

Tab. 4.9: Performance indicators of tested models trained on data sets augmented by different intensity shifts. Transformations were applied to polar representation. Training on polar (Pol) and cartesian (Car) representation is considered.

Repr.	DA Technique	SENS	SPEC	PPV	BACC	F1-score	MCC
Pol	no DA	**0.6870**	**0.5838**	**0.7737**	**0.6354**	**0.7171**	**0.2697**
	Tunica Variation	0.6644	0.5928	0.7717	0.6286	0.7067	0.2413
	Column Scaling	0.5409	0.7480	0.7986	0.6444	0.6650	0.2234
	Random Guidewire	0.6373	0.6281	0.7778	0.6334	0.7003	0.2307
	Blood Speckle	0.6900	0.5719	0.7707	0.6309	0.7165	0.2634
Car	no DA	**0.7068**	**0.5480**	**0.7589**	**0.6274**	**0.7266**	**0.2544**
	Tunica Variation	0.6872	0.5594	0.7589	0.6240	0.7202	0.2289
	Column Scaling	0.5637	0.7147	0.7858	0.6399	0.6785	0.2111
	Random Guidewire	0.6570	0.5923	0.7630	0.6254	0.7098	0.2154
	Blood Speckle	0.6885	0.5778	0.7722	0.6331	0.7168	0.2666

The last DA techniques tested are curve variation, columns scaling as well as random guidewire creation and random blood speckle imitation. For all transformations, different variations were tried and representative test results were listed in tab. 4.9. Curve variation was varied in the intensity of the maximum possible curve shift. A maximum of one additional guidewire was inserted per image to avoid covering larger sections of image content.

Data augmentation with artificial artifacts does not boost bacc and mcc in any of the cases. Artifacts that affect larger areas of pixels and change their contrast ratios deteriorate the test results in particular more than 4%.

4.4 General Remarks

In most cases, the effects of the various DA techniques do not differ noticeably when the images were transformed into cartesian format. Exceptions are all transformations that process the margins of the image unevenly, causing prominent artifacts after its cartesian transformation. In this case, the corresponding transformation should be omitted, or training on the polar representation should be preferred. An example of this is horizontal shearing of the B-scan on the temporal axis. Scaling shows a similar effect.

In general, in our experiments models trained with polar images tend to perform better. In addition, it was observed that DA techniques strongly help to determine classes that are unevenly weighted and difficult to predict. Furthermore, it was observed that DA techniques strongly help to determine unweighted and harder-to-predict classes. Sens and spec are well suited indicators for this phenomenon. It was observed that DA techniques strongly help to determine unweighted and harder-to-predict classes. For this phenomenon, SENS and SPEC are good indicators. The harder to predict class benefits significantly from data augmentation, while the complementary value suffers slightly. Nevertheless, the total MCC and BACC are significantly increased.

5 Discussion

The main objective of this bachelor's thesis was to develop a pipeline that allows qualitative prediction performance measurement of trained neural networks. The second objective was to investigate the effect of partially new DA techniques when training models with IVOCT data sets.

5.1 Proof of Concept

There were three requirements that the pipeline had to meet in order to fulfill the main objective of this work. First, a model designed for the specific purpose had to be assembled, which means meeting the current state of the art requirements for medical image recognition and finding appropriate hyperparameters to make comparable assumptions for later knowledge transfer. Next, established as well as new pre-processing and augmentation techniques had to be implemented. Third, various state-of-the-art metrics were implemented to monitor and evaluate training processes and test performance. In the following the fulfillment of these requirements is discussed based on the conducted experiments on our pipeline with IVOCT sample data.

5.1.1 Model Design

During the experiments it had to be determined that accuracy is only a very limited measure to select the best model for the performance tests with the help of validation, when training on small IVOCT data sets. A major indicator of this was the test results themselves. To keep track of potential inconsistencies, we tested both the most recently trained model and the model deemed best by the early stopping procedure for each test. Depending on the hyperparameters, in particular the maximum number of epochs, the early stop rate, and the initial learning rate, we observed that the testing performance measures vary greatly between the best and the last training epochs. Furthermore, extremely divergent test results arise for each cross-validation fold. One reason for this could be the way Accuracy itself is computed, which does not take into account the imbalance of the data set and extreme bias of its subsets. In addition to a potentially unfortunate data split, the divergence and inhomogeneity of the data set is considered another reason. As a result, the choice of the best epoch depends immensely on the nature of the validation set. For this reason simply halving the initial learning rate, which reduces the fluctuation and thus enables a more precise choice of the tested model, does not address this effect appropriately. K-fold cross-validation with its associated mean computations could additionally amplify that effect. Since the test set can differ strongly from the validation as well as from the training set, extremely different test results may arise for each fold which makes comparison of single folds meaningless. For example, a high ACC of about 74% is sometimes achieved together with a SENS of over 90%, while the SPEC is around 50%. Put simply, if a network is trained with 60 related images and 40 other unrelated images, a model trained on a joint set would likely classify most of the images correctly. In contrast, if two different models are each trained on a subset, they would likely classify the images in a ratio of 60 correct and 40 incorrect images. Nevertheless, due to the strong bias, CV was essential for our experiments, since the test results of the individual folds partially differed substantially from each other.

Replacing BACC with the MCC as a measure for comparing individual folds resulted in a more uniform and precise selection of validated epochs, but did not contribute sufficiently to overcome the problem of high diversity in the data set. In our particular early stopping algorithm an additional criterion was used to trigger the mechanisms. Each epoch is tested not only on the validation set, but also on equally distributed parts of the training set. If the performance of the model applied to the validation set increasingly exceeds that of the test set, this is an indicator of overfitting and the early termination mechanism was triggered. For relatively homogeneous data, this method works very well. During the development, however, it had to be determined that the method is not suitable for small, highly diversified IVOCT datasets. Due to the bias, the model sometimes predicts significantly better on either test and validation data in all almost cases. Thus the early stopping criterion may be permanently fulfilled, regardless of how the models performance behaves compared on the two sets. Therefore, we omitted this additional criterion. Still, the choice of epochs to test still remained too noisy. Hence early stopping was omitted and a maximum number of 30 training epochs was set equally for each fold after comparing their individual validation performance. We set an initial learning rate of 5e-7 to reduce fluctuation in some specific folds during training. Due to this choice, the model tends to exhibit symptoms of overfitting during training. However, this is relativized to some degree by the use of data augmentation. Since optimisation of performance itself is not objective of this work, we tolerate the over- or underfitting that might occur for each epoch. It is noted that this can have a major influence on the overall performance evaluation. Nevertheless, the methodology is sufficient to clearly identify and investigate the general impact of DA techniques on IVOCT data.

5.1.2 Augmentation Impact

Classical data augmentation techniques have, according to the results, a greater positive effect on the performance of the corresponding generated models than IVOCT specific techniques that only modify smaller sections of the image. Considering that with small IVOCT data sets there is only a very limited possibility for fine tuning, it appears to be a logical conclusion that positional information about desired features, especially with very diversified data, matter more than their nature and robustness to potentially interfering artifacts. This is consistent with the observations that less distinct images with inherently lower cross entropy containing less new information for the network, result in little improvement in performance especially at earlier stages of training. Thus, such techniques are preferred for the optimization of networks trained on less diverse data sets.

Underlying Model

As the amount of data increases, it is assumed that overfitting occurs later during training. This is caused by the fact that the error can only be reduced slower when diversifying transformations are applied and the learning rate is adjusted correspondingly.

The effect of an decreased deviation of the validation performance when testing on the same data set with and without DA supports this statement (see fig. 4.4 and fig. 4.3). This may lead to a horizontal shift of the curve center in the gridsearch graph. In order to ensure a certain tolerance of the model, the midpoint therefore seemed to be particularly suitable for initial tests. In order to ensure a certain tolerance against changes in the data set, the center point for tests seemed to be particularly suitable. Inspecting the curves of training loss plotted against the overall batch number while training, supports our choice (see fig. 4.2.

For the smallest of our tested learning rates an early stopping accuracy of 1e-7 suffices, since we observe mostly larger deviations after reaching the maximum of their validation measures. We choose this value in assumption that future training processes do not converge with a smaller rate to their validation accuracy maximum.

Comparing the presented values of the listed initialization and pre-training variants in tab. 4.1, reveals that the performances of the models vary considerably. Significantly different performance values between CVs in a single training indicate the missing ability to generalize features. For this reason, no meaningful tendencies or rules can be derived, when training non-augmented data, even though CV was applied. Transfer learning is used in all further experiments, since in general no worse values can be expected from pre-training than from random initialization of the weights and biases. Rescaling on the other hand ensures a uniform floating point representation.

Without CV no meaningful comparison would be applicable, because the individual folds (see fig. 4.3) deviate to much when considered individually. The performance depends fundamentally on each validation or test set and the epoch from which the model is chosen. This effect is particularly noticeable, when sensitivity and specificity are considered individually.

A problem when finding the right hyperparameters for scaling, transfer learning and padding the grayscale image is, that validation performance fluctuates excessively, when no DA is used on small data sets. Therefore it may give less representative results for fine tuning. One way to counteract this would be fine tuning the hyperparameters after the data have been augmented and simultaneously adapt the initial learning rate. Alternatively one could adapt the initial learning rate for every CV step individually.

Affine Transformation

Due to the nature of IVOCT, the model benefits from further generalization regarding the horizontal position of the features. Intuitively spoken the horizontal position does not seem to be of particular relevance for feature classification. Thus rotation or horizontal shifts can reduce the dependence on certain positions of the features. This is also reflected in our results since both transformations have a positive impact on the models performance. Larger vertical shifts additionally require the network to filter out the otherwise rather statically positioned features like catheter reflections, which increase the complexity of learning from small data sets. In most cases vertically shifted images cannot replace realistic images. It was therefore expected that shifts tied to the authentic positions of features are more effective for training with small IVOCT data sets. Our expectation that vertical shifts would therefore result in a smaller overall increase in performance than horizontal shifts was not fulfilled. This indicates that the model can easily filter out certain artifacts, such as catheter reflections, than it can learn the desired feature properties. The bias of the testing set may even reduce the MCC score slightly. We conclude that the value of repositioning the desired features is higher than the different positioning of the artifacts. However, unproportionally boost could not be identified when training on smaller shifts. For this reason, Davella's proposal to give more weight to smaller shifts [25] is rejected when training on IVOCT data. Since we could not show any measurable improvement, when training on IVOCT data with both, fully horizontal and vertical shifts, we make use of fully horizontal shifting and only little vertical shifting of up to 20 pixels for our strategy only.

It is not possible to draw conclusions about the development of individual test results from an increase in overall performance through stretching, shearing or scaling, as these show both positive and negative trends when compared with the MCC and BACC scores. In principle, it can be deduced from the observations of the tests with stretching that it has a positive effect on performance. Shearing on the other hand shows a higher MCC score only when training with polar representations, which could be related to the fact that additional artifacts are generated in this format. Combined with the fact that shearing also removes information from middle rows, this limits the learning effect for the model.

Elastic Deformation

The fact that the performance improvement due to augmentation with elastic deformation is not as large as when the models are trained on cartesian representation may have several reasons. In polar representation all sections are distorted uniformly according to a Gaussian filter. The cartesian transformation is not taken into account, which is why the deformations decrease towards the center of the image. They are amplified with increasing distance to the center. In the case that the elastic deformation is applied after the cartesian transformation, the edges of the circle are deformed as well. This represents a large discrepancy to the realistic images and can also have a negative influence.

Intensity Shifting

Random brightness changes of the image are similarly caused by the pre-processing transformations. Small, particularly bright maxima are moderated by bilinear interpolation, since these often affect very few pixels that were averaged with others. This is most likely the reason why an additional random but small brightness changes have only a relatively limited effect. It should be mentioned that interpolation alone is not responsible for the brightness change of the entire image. This is only due to the fact that interpolation reduces maximum values and the subsequent rescaling therefore illuminates the entire image. When examining the histogram, it can also be seen that the brightness of the images is relatively evenly distributed. The added value of augmentation by brightness jittering is therefore very limited anyway.

Noise Filters

The observation that data augmentation by adding noise does not result in a detectable increase in performance is in line with our expectations. The model can derive the most important information in IVOCT images using deeper structures. For example, to distinguish the arterial wall or the lumen, it is usually sufficient to measure a regional contrast ratio. For this reason, fine structures seem to play only a minor role in classification tasks of IVOCT images. Various noise filters protect especially the first convolutional layers against overfitting. This limits the effect that artificial noise can have on IVOCT recordings. Each time noise is applied, new images are created from which the model could learn. However, the model benefits from this only to a limited extent. In this context, it is important to note that the images come from different pullbacks that are related in content. Two images taken one after the other show only few differences and usually have high mutual information. However, they have different noise sources, which means that the model already learns the natural noise distribution of the images. Additional noise that has a Gaussian distribution or changes only individual pixels, such as salt and pepper noise, brings a barely measurable advantage. At particularly high intensities, the augmented images deviate too much from the originals, so that classification becomes less effective.

Partial Masking

Partial masking mainly contributes to better generalizability. Since it does not improve performance, we conclude that the model already prevents overfitting well by methods like pruning or L2 generalization with similar methods well enough. The extra cost of partial masking and guide wire removal can thus be omitted when training with an advanced pipeline. The latter approach could be even more effective. The width of the guidewire artifacts are slightly different. Possibly little but valuable information is lost due to the partial masking.

Artefact Imitation

Artificial DA techniques like random distortion, curve variation, column scaling and blood speckle appear almost like realistic images. Only the creation of random guidewires clearly distinguishes an image with features from the real thing. Augmenting the data with our artificial artifacts decreases the performance of the resulting model. We find that the model loses performance when artificial transformations are applied to the original image. This can be due to several reasons. First, there is an imbalance of shown anatomies in the data and their nature allows artificial artifacts to be inserted only in certain locations. The model learns prominently about their position and presence in combination with the associated labels. It does not learn the ability to examine the images for the targeted features, which leads to frequent missclassification and high bias. This can be observed in curve variation and random guide wire augmentation. Column scaling, blood speckle and random distortions are always randomly arranged in the image. This is also reflected in the results. They have only a slight negative influence on the test results. The model learns to deal with the additionally inserted artifacts, but does not benefit from them.

We conclude that, in general, transformations for which we cannot detect significant differences from real images when assessed individually by eye are preferable for augmentation. Also the tested artefacts inserted independently of the images content do not fit well enough such that it helps the model to generalize better. Because of the high complexity to artificially recreate artefacts it is recommended that any augmentation strategies are restricted to simpler techniques. They concentrate on commonly used transformations, e.g. affine transformations and elastic deformations. In particular, the training process does not benefit from artificial artefact insertion.

5.1.3 Performance Assessment

In medical diagnostics, SENS is the ability of a test to correctly identify individuals with the disease, while specificity is the ability of the test to correctly identify individuals without the disease. If the SENS is high, any person who has the disease is likely to be classified as positive. If, on the other hand, the SPEC is high, any person who does not have the disease is likely to be classified as negative. We choose precision when we want to have greater confidence in true positive diagnoses and rather diagnose some patients false positive than causing some false negative diagnoses missing out on some injured patients.

In our tests, we assign the same importance to the detection of diseases as to ruling them out. In medical diagnostics a different weighing may be desired. For instance, a false-positive sample classification is often tolerable to some degree, unlike an undetected, misdiagnosed stenosis that may cause greater harm to the patient. For this, a greater weight could be assigned to recall. If, on the other hand, the priority is to avoid false positives, specificity is to be weighted higher. Weighting in the model can be achieved by using weighted metrics or intentionally unbalanced data sets while training.

Even though by BACC and MCC the performance discrepancies of the models could be evaluated meaningfully, it must be noted that our models shows difficulties in detecting true negatives. This could also be caused by the composition of the test set and the diversity of the data. Increasing preciseness in measurement may be reached, by manually observing and choosing pullback frame series for manually setting up a test set with balanced labels. A second major influence might be the comparatively low depth of our model. With additional convolutional layers, the model would be able to learn more complex features and thus less frequent features, which would allow the underrepresented negative samples to be classified with greater certainty.

5.2 General Remarks

We conclude our discussion with some general remarks on the limitations and feasibility of using data augmentation and pre-processing in real-world IVOCT classification applications. Fundamental limitations of Deep Learning concerns the precision of diagnosis by labeling specialists, generally determined by the majority of multiple doctors. Another critical aspect is that the CNNs cannot diagnose unusual diseases that were not included in the original database. [61] Noisy labeling might be caused by tissues which are difficult to distinguish and thus potentially degrade the models performance [57].

Although the overall classification accuracy of in our experiments reaches approximately 80%, learning methods applied to IVOCT could lead to clinically useful results. During intervention, a cardiologist is interested in plaque deposits rather than in reviewing individual frames. To apply the appropriate plaque modification strategy and decide on appropriate stent size, the interventional cardiologist must identify large circumferential calcifications that might hamper stent deployment and lipidous lesions [57]. Thus, high resolution is not required for any of these scenarios. Processing (including pre-processing and classification) time with less than 0.4 seconds suggests that live-time clinical usage is possible. [57] In this sense, the results strongly encourage further research to improve IVOCT scan classification.

6 Outlook

6.1 Model Design

In general, ResNet-18 is a relatively small and compact neural network that has limited ability to learn more complex functions. Other models like VGG net or AlexNet could potentially benefit more from data augmentation [36, 67, 108]. Other hyperparameters, the loss-function or pruning algorithms might help the model to generalize better.

Since our experiments were strongly influenced by the diversity and bias of the data set, it may be useful to test whether the collected results apply equally to other data sets. The use of higher resolution images may affect the experimental results as well. Comparable data sets may also be used for transfer learning to finetune the model and address unanswered questions.

To conduct our experiments, we apply relatively simple criteria to select the model from the training process to run on our test set. The main criterion for early stopping was the validation performance with the MCC score. As another criterion, we tested the comparison between the validation and the test performance. These did not lead to meaningful results in our experiments, such that only models of manually selected epochs could be compared. In the future, more advanced algorithms could be applied for choosing the right model with early stopping. This would inevitably lead to improved efficiency due to the saved computing time in the training process and increased performance. Stochastic weight averaging is one example to reach higher performance of well-tuned models.

6.2 Data Augmentation

Artificially recreating complex artifacts to augment data is a very complex and cost intensive process. For this reason, it is only possible and useful to imitate them realistically to a certain degree. In this work emulated transformations were built to better generalize the model. However, for various reasons, these methods did not have a positive impact on the test results when trained with the IVOCT dataset and did not help to generalize the model. Since tunica variation and artificial blood speckle did not deteriorate the performance as much, there might be potential in continuing further development. An unexplored method is the use of GANs, which can better extract the properties of features and artifacts, and thus could potentially achieve much more realistic replicas of IVOCT images. Antoniou [3] and Odaibo [70] did further investigations with a detailed focus on OCT imaging.

Elastic deformations could be improved by excluding certain mostly static or unimportant sections like catheter reflections from them leading to a more realistic output [84]. For example, it could be avoided that circle edges are deformed in the cartesian representation. In addition, a more realistic image could be generated by applying a uniform deformation after the cartesian transformation, which in turn could have a performance enhancing effect.

Various tests on adding noise show no or negative effect on the MCC and BACC. Automatic, conditional and more accurate assessment of the realistic noise distribution could make a more accurate recreation possible, so that the model can benefit from the noise filters.

Future studies may determine in detail if and how much the combination or concatenation of different DA techniques affects the overall models peak performance. For instance it may be useful to further investigate whether vertical shifting during training leads to improvement or deterioration when combined with horizontal shifting. Further investigation may give insights in the ability of generalization when training on vertically shifted images.

In general, it is questionable whether all techniques with a positive effect still offer added value in the finetuning of the model. This question arises especially with techniques that generate images that deviate strongly from real images.

7 Conclusion

In this work DL models were developed that considered the detailed use of different DA techniques specifically for IVOCT data sets for the first time. The pipeline contained a data loader transforming the B-scans online with different techniques substituting the effect of actual augmentation. The iterative development of different models with their automatic performance evaluation helped to find valuable transformations for building an effective augmentation strategy. In contrast to previous investigations, we presented a variety of diverse methods that proved the concept of augmenting medical image data for training NNs. The application of the techniques was demonstrated on the implemented pipeline utilizing ResNet-18 in its core. The augmentation of medical IVOCT data with various artefacts demonstrated the performance boost that can be achieved by building an appropriate strategy combining multiple techniques. The trained models therefore come with measurable performance enhancement that could aid in developing a system providing automatic decision support and analysis of the in vivo vascular anatomy. It was shown during augmentation experiments that the different techniques provide varying enhancement degrees of model performance. CV allowed the comparison of different runs and enabled us to deal with strong fluctuation and different behavior of each fold. The enhancement estimation based on individual techniques was often volatile and significant distinctions were only measurable when combined for training a single model. Techniques that were found to have positive effect on the models classification performance were compared and proposed for further investigation. Hereafter, domain specific training strategies may be build for future fine tuned training. It was shown that the models performance is determined by the extent to which relevant information and features of the original medical images are preserved in the augmented training sets. Overall, we find that a model trained with data augmented by the discussed techniques, e.g. affine transformations, may boost the performance by 0.05 in Matthews correlation coefficient (MCC) and 0.03 in balanced accuracy (BACC). Less effective transformations such as contrast jittering may deteriorate scores significantly by more than ten percent. Obtained results proved the immense impact and therefore the relevance of DA when building deep learning models with the use of small IVOCT data sets. The implemented pipeline and transformations may now serve as a basis for future investigations into DA and development with further fine-tuning in training with predictive performance satisfactory for clinical use.

A Appendix

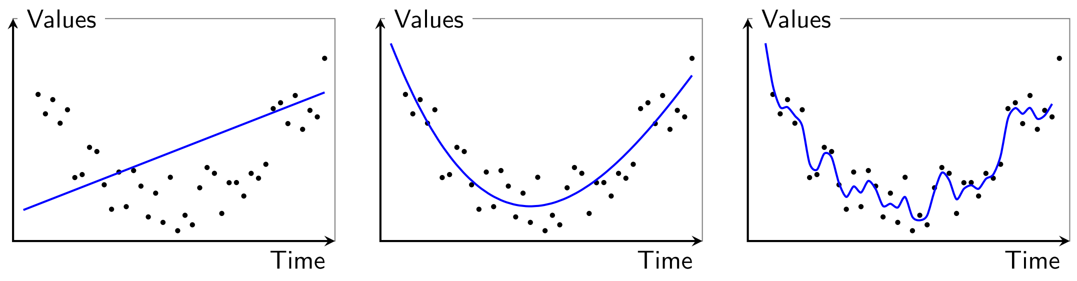


Fig. A.1: Comparison of under- and overfitting (left and right) to a balanced well generalizing model (center). [94]

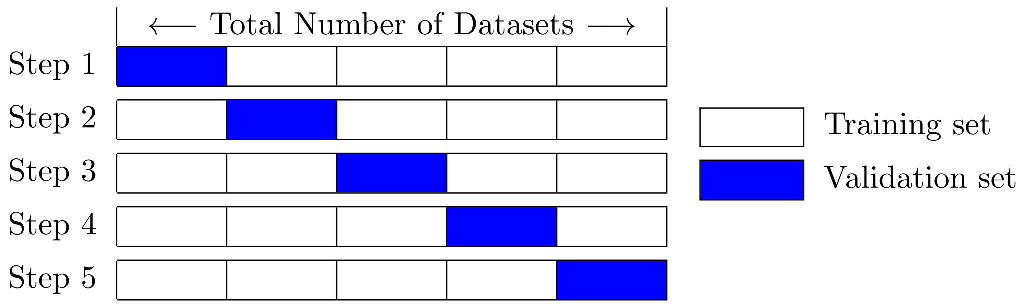
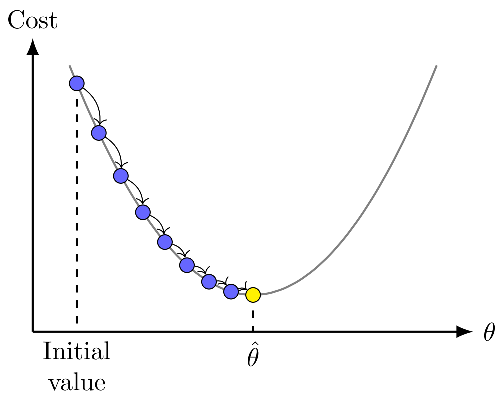
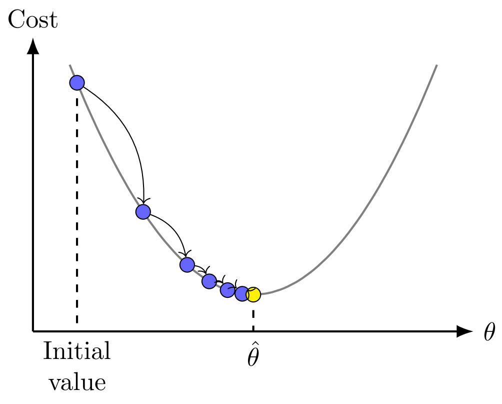


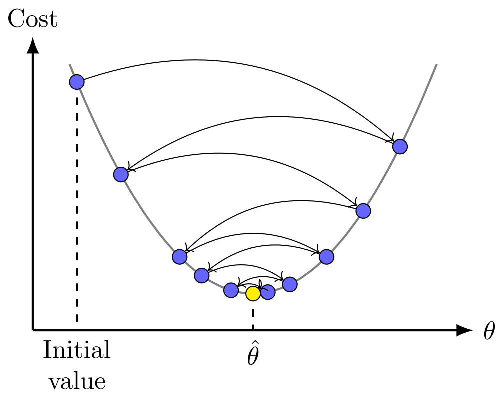
Fig. A.2: Subdivision of known data for training (test data excluded). The validation data (blue) are used for error estimation while training with the training set (white). For each step there is a model trained and tested. [87]



(a) Small Learning Rate



(b) Medium Learning Rate



(c) High Learning Rate

Fig. A.3: Comparison of different initial learning rates for training machine learning models. Plotting with function gradient and local minimum in the center (yellow). [101]

B Appendix

Tab. B.1: Architecture of ResNet-18 in a layered tabular description summarizing the convolutional functions and connections of the classification model. [80]

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	3×3,64, stride 2
conv2	$56 \times 56 \times 64$	3×3 max pool, stride 2 $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	N	

C Appendix

Pipeline: Impact of different Data Augmentation Techniques for Deep Learning with Optical Coherence Tomography (IDDATDLOCT)

"Impact of different Data Augmentation Techniques for Deep Learning with Optical Coherence Tomography" is the topic which is investigated as well as discussed in the bachelor thesis by Lennard Korte at the Institute of Medical Technology and Intelligent Systems (MTEC) at Hamburg University of Technology (TUHH). This repository includes the complete academic work and the code basis whose usage is explained below.

Table of Contents

Quickstart Guide for Linux

Note: Training data for IVOCT are not provided in this repository due to data protection reasons.

1. Clone the Repository to your home directory
2. Add training data under: `/<home_directory>/IDDATDLOCT/data/`
3. Install Docker and NVIDIA Container Toolkit
4. execute training and testing

```
$ bash exe/run_train_and_eval_docker.sh
```

Quickstart Guide for MTEC Irmo Server

Note: A copy of this repository including training data
is stored on the irmo server.

1. Connect via VPN: (Website Rechenzentrum TUHH)
2. Run Training and Testing on Irmo Server
   1. Login to irmo server via remote SSH repository Access:

      ```
      $ ssh <username>@irmo.et8.tu-harburg.de
      ```

   2. Mount Pallando Volume:

      ```
      $ gio mount smb://pallando.local/share/
      ```

   3. Access repository:

      ```
      $ cd /run/user/1000230/gvfs/smb-
      share:server=pallando.local,share=share/students/Korte/IDDATDLOCT
      ```

   4. Copy the training data (or alternatively the entire repository) to your home directory on the irmo server:

      ```
      $ cp -r ./data/ /$HOME/IDDATDLOCT/data/
      ```

   5. Run training and testing script:

      ```
      $ bash exe/run_train_and_eval_docker.sh
      ```

System Requirements

- either or:
  - Windows 10 build is 17063 or later
  - MacOS Mojave or later
  - Ubuntu 20.04 or later
- required:
  - NVIDIA Graphics Card
  - Minimum 16GB RAM
- recommended:
  - 250 GB SSD Storage

Academic work

See Academic work

Features

This pipeline provides a simple way of training and testing deep learning models. It offers a number of options to customize the training processes. There are checkpoints and logs to observe and monitor more of its details. A number of parameters may be adapted in a specified configuration file or in the code itself to customize the pipeline in almost any way, i.e. add methods for data augmentation or imege preprocessing techniques. The pipeline can additionally generate examples of the processed images to visualize modifications on images by the human eye. More detailed information can be found in the academic work linked above or in the usage description below.

Logging

For every Cross-Validation Iteration there is an new folder under the given name. The training output and progress is protocolled in the specified directory under `training.log`. Tests results will be stored in the specified working directory under `test_results.log`. Additional Logging and visualization can be enabled by providing a W&B API key as an argument and activating W&B in the configs.

Checkpoints

The current model is saved in `latest_checkpoint.pt` after each epoch, while finished training progress is indicated by a renaming to `last_chackpoint.pt`. Checkpoints with the best validation accuracy are preserved in `best_checkpoint.pt`. The checkpoints will be saved under `./data/checkpoints/projectspecific/group/name/`.

Note: checkpoints contain:

```
{
'Epoch': epoch,
'Model': self.model.state_dict(),
'Optimizer': self.optimizer.state_dict(),
'Scaler': self.scaler.state_dict(),
'Wandb_ID': self.wandb_id
}
```

Folder Structure

```
IDDATDLOCT/
│
├── academic work/        - Bachelor Thesis and defense
│
├── data/                 - holds large data files like training data and DA
Examples
│
├── exe/                  - holds executable bash and batch files
│
├── src/                  - holds all python source code and standard configuration
│   ├── checkpoint.py             - defines training checkpoints with models,
optimizer, etc.
│   ├── config_standard.json      - specifies the standard configuration of
application
│   ├── config.py                 - evaluates configurations, arguments and
provides Config object
│   ├── create_samples.py         - enables creation of samples of images
│   ├── da_techniques.py          - implements data augmentation techniques
│   ├── data_loaders.py          - defines dataloaders for training, validation
and testing
│   ├── dataset_preparation.py   - prepares the dataset for the dataloaders
│   ├── dataset.py               - represents the dataset object to dataloader
│   ├── eval.py                  - calculates metrics to assess performance
│   ├── logger.py                - for logging result files and printing tasks
│   ├── main.py                  - entrance point for application
│   ├── models.py                - defines all models to train on
│   ├── train_and_test.py        - Main training loop file
│   ├── utils_wandb.py           - Wandb logging class
│   └── utils.py                 - Helper functions and utilities
│
├── tests/                - holds all tests and corresponding configurations for
academic work
│
├── .dockerignore         - manages build files for docker
│
├── .gitignore            - specifies intentionally untracked files Git should
ignore
│
├── colab_setup.ipynb     - Notebook file for Colab setup
│
├── config.json           - specifies all non standard configurations
│
├── Dockerfile            - contains all the commands to assemble an image
│
├── initial inspiration   - code this project got inspired by
│
├── LICENCE               - contains legal notice
│
├── requirements.txt      - contains all modules required for applications
execution
│
└── ...
```

Usage

Note: All commands in the following are intended to be executed in the bash command line.

Prerequisites / Installation

Check if NVIDIA graphics card is available with:

```
$ ls -la /dev | grep nvidia
```

For running locally

1. Install Python
   - Windows: Install Python via Microsoft Store.
   - Ubuntu:

```
$ sudo apt-get update

$ sudo apt-get install python3.9
```

- MacOS:

```
$ /bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

$ brew install python3
```

2. Install Pip

```
$ curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

$ python3 get-pip.py
```

3. Install further requisites

```
$ pip install -r requirements.txt
```

For running in Docker

Note: NVIDIA Container Toolkits are Linux only drivers. Therefore containerization as well as this guide is Linux only.

1. Install Docker

```
$ curl -fsSL https://get.docker.com -o get-docker.sh

$ sudo sh get-docker.sh
```

2. Install NVIDIA Container Toolkit

```
$ distribution=$(. /etc/os-release;echo $ID$VERSION_ID) \
    && curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key
add - \
    && curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-
docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list

$ sudo apt-get update && sudo apt-get install -y nvidia-docker2

$ sudo systemctl restart docker
```

3. Availability of GPU's in Docker can be testes with the following command:

```
$ sudo docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi
```

For running in Colab

To run the application in colab, setup the mounted machine with the provided commands in the `colab_setup.ipynb`-file. Git access rights to this repository are required to follow these steps. Alternatively the repository can be copied to the mounted Google-Drive folder manually. The application is eventually ready to run the same way as a local run.

Training

Note: When training has been interrupted, it will be continued from the last checkpoint with the old configuration. A warning will be shown before continuing training in case the configuration changed. The W&B API key as an argument is only required, when W&B is enabled.Furthermore you can view the program manual by calling the program with the help flag, e.g.: `bash exe/run_train_and_eval_docker.sh --help`

Run locally

- Windows:

  ```
  $ exe/run_train_and_eval_local.bat -w <WANDB_API_KEY>
  ```

- Mac / Ubuntu

  ```
  $ bash exe/run_train_and_eval_local.sh -wb <WANDB_API_KEY>
  ```

Run in Docker

(Linux only)

```
$ bash exe/run_train_and_eval_docker.sh -wb <WANDB_API_KEY>
```

Testing

The application tests the trained model automatically after it has been training with the best validated model and the last trained model.

Data Sampling

The Pipeline can automatically generate a small number of samples that are output in '.*h5'- and '*.jpg'-format. For this the flag `--show_samples` has to be added as described above. These samples are equally distributed over the training data set and stored in the data directory under `./data/h5s/data_augmentation/` and `./data/jpgs/data_augmentation/`. They are brightened and so may help to illustrate the preprocessing and data augmentation techniques.

Configuration

Arguments

The configuration of the application can also be changed via command line arguments specified directly, when starting the program, e.g.:

Activate Wandb statistics upload

```
$ bash exe/run_train_and_eval_docker.sh -wb <WANDB_API_KEY>
```

Deactivate Training and Testing

```
$ bash exe/run_train_and_eval_docker.sh -ntt
```

Activate sample data augmentation

```
$ bash exe/run_train_and_eval_docker.sh -smp 2,3
```

Choose Devices for multi GPU training

```
$ bash exe/run_train_and_eval_docker.sh -gpu 1,2
```

Choose configuration file

```
$ bash exe/run_train_and_eval_docker.sh -cfg ./config.json
```

Choose indices of image transformations to be used in the specified order

```
$ bash exe/run_train_and_eval_docker.sh -da [0,2]
```

Choose configuration file

```
$ bash exe/run_train_and_eval_docker.sh -ycf ./config.json
```

Configuration File

Configurations are copied and stored in the `name/` directory after starting the application for protocolling purposes. Different configurations may be provided via config file in `.json`-format under the path (`any_dir/config.json`) given by argument. When using docker the directory for the configuration file must be `./config.json`. Only configurations that have to be changed need to be specified. The standard configuration looks like this:

```
{
    // Parameters for testseries and research
    "name": "run0",
    "group": "group0",

    // Hardware
    "use_cuda": false,
    "deterministic_training": true,
    "deterministic_batching": true,     // Only effective if deterministic training
is used

    // Model
    "model_type": "ResNet18",
    "pretrained": true,
    "num_out": 2,
    "loss_function": "cross_entropy",
    "optimizer": "Adam",

    // Data subdivision for Cross-Validation
    "set_percentage_cv": 80,            // Size of training set in percentage
    "set_percentage_val": 20,           // Size of validation set when num_cv = 1

    // Dataloader
    "preload": false,
    "batch_size": 128,

    // Wandb
    "enable_wandb": false,
    "send_final_results": true,
    "b_logging_active": false,

    // Logging
    "calc_train_error": true,           // (If calculated also used in early
stopping)
    "peak_train_error": false,
    "calc_and_peak_test_error": false,

    // Training
    "num_cv": 7,                        // If 1: val set size is
set_percentage_val, if >1: (training set size)/num_cv
    "epochs": 50,                       // Maximum number of epochs
    "learning_rate": 3e-6,
    "early_stop_patience": 15,          // Number of epochs to stop after when ES
condition true, 0 if no ES
    "early_stop_accuracy": 7,           // Digits accuracy is rounded to
    "transformations_chosen": "",       // List of Indices of applied
transformations

    // Dataset specifics
    "define_sets_manually": true,
    "c2_or_c3": "ivoct_both_c2/",       // Specify on which data to train on
    "cart_or_pol": "orig"               // "orig" chooses cartesian format as
source. "pol" uses polar representation source
}
```

Customization and Modification

Custom CLI options

If some configurations need to be changhed often or quickly, then it is usefull to have command line options. By registering custom options as follows you can change some of them using CLI flags.

```
CustomArgs = collections.namedtuple('CustomArgs', 'flags type target')
options = [
    CustomArgs(['--lr', '--learning_rate'], type=float, target=('learning_rate')),
    CustomArgs(['--bs', '--batch_size'], type=int, target=('batch_size'))
    # Add more custom args here
]
```

`target` argument is sequence of keys, which are used to modify that option in the configuration. In this example, `target`
for the learning rate option is `('learning_rate')` because `config['learning_rate']` points to the learning rate.
`bash exe/run_train_and_eval_docker.sh -c config.json --bs 256` runs training with options given in `config.json` except for the `batch size` which is increased to 256 by command line options.

Update Requirements File

In case any changes were made to the code affecting the imports of the application, the requirements file can always be updated (or replaced in case there is one already) by generating a new requirements file with:

```
$ pip3 install pipreqs

$ pipreqs --savepath ./requirements.txt ./src
```

Note: All modules have to be installed for this command to work propperly which are supposed to be imported (unexpected behaviour).

License

This project is licensed under the MIT License. See License for more details

Acknowledgements

This project is inspired by the code base provided by Robin Mieling from the MTEC institute. Almost all code has been reqritten, but many general concepts remain the same.

Bibliography

[1] Abdolmanafi, A., Duong, L., Dahdah, N., Cheriet, F.: Deep feature learning for automatic tissue classification of coronary artery using optical coherence tomography. Biomedical optics express 8, 1203–1220 (2017)

[2] Albawi, S., Mohammed, T.A., Al-Zawi, S.: Understanding of a convolutional neural network. International Conference on Engineering pp. 1–6 (2017)

[3] Antoniou, A., Storkey, A., Edwards, H.: Data augmentation generative adversarial networks, http://arxiv.org/pdf/1711.04340v3

[4] Bahoshy, L.P.: What is optical coherence tomography (oct)? 3d digital eye exams (2022), https://stoneycreekeyecare.com/what-is-optical-coherence-tomography-oct/

[5] Barron A., R.: Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information Theory 39, 930–945 (1993)

[6] Beaurepaire, E., Boccara, A.C., Lebec, M., Blanchot, L., Saint-Jalmes, H.: Full-field optical coherence microscopy. Opt. Lett. 23 pp. 244–246 (1998)

[7] Beers, B.: Regression definition: What is regression? (2021), https://www.investopedia.com/terms/r/regression.asp

[8] Ben-Cohen, A., Klang, E., Amitai, M.M., Goldberger, J., Greenspan, H.: Anatomical data augmentation for cnn based pixel-wise classification. IEEE 15th International Symposium on Biomedical Imaging (ISBI) pp. 1096–1099 (2018)

[9] Bille, J.F.: High resolution imaging in microscopy and ophthalmology: new frontiers in biomedical optics. Springer (2019)

[10] Bioucas-Dias, J.M., Figueiredo Mário A. T.: Multiplicative noise removal using variable splitting and constrained optimization. IEEE Transactions on Image Processing 19 pp. 1720–1730 (2010)

[11] Boyko, E.J.: Ruling out or ruling in disease with the most sensitiue or specific diagnostic test: Short cut or wrong turn ? Medical Decision Making 14 pp. 175–179 (1994)

[12] Brownlee, J.: Gentle introduction to the adam optimization algorithm for deep learning (2017), https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[13] Brownlee, J.: Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions. Brownlee, Jason (2019), https://machinelearningmastery.com/better-deep-learning/

[14] Campos, G.F.C., Mastelini, S.M., Aguiar, G.J., Mantovani, R.G., de Melo, L.F., Barbon, S.: Machine learning hyperparameter selection for contrast limited adaptive histogram equalization. EURASIP Journal on Image and Video Processing (2019)

[15] Cao, C., Chicco, D., Hoffman, M.M.: The mcc-f1 curve: a performance evaluation technique for binary classification. Computer and information sciences, I.2.0, 68T05 arXiv (2020), `https://arxiv.org/abs/2006.11278`

[16] Celi, S., Berti, S.: In-vivo segmentation and quantification of coronary lesions by optical coherence tomography images for a lesion type definition and stenosis grading. Medical image analysis 18 pp. 1157–1168 (2014), `https://www.sciencedirect.com/science/article/pii/S1361841514001054`

[17] Chen, B., Cai, J.L., Chen, W.S., Li, Y.: A multiplicative noise removal approach based on partial differential equation model. Mathematical Problems in Engineering pp. 1–14 (2012)

[18] Chicco, D., Jurman, G.: The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. BMC genomics 21 p. 6 (2020)

[19] Chicco, D., Starovoitov, V., Jurman, G.: The benefits of the matthews correlation coefficient (mcc) over the diagnostic odds ratio (dor) in binary classification assessment. IEEE Access 9 pp. 47112–47124 (2021)

[20] Chicco, D., Tötsch, N., Jurman, G.: The matthews correlation coefficient (mcc) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. BioData mining 14 p. 13 (2021)

[21] Cho, J., Lee, K., Shin, E., Choy, G., Do, S.: How much data is needed to train a medical image deep learning system to achieve necessary high accuracy? ICLR (2015)

[22] Community Editors: Opencv 2.4.13.7 documentation (2022), `https://docs.opencv.org/2.4/`

[23] Cruz-Roa, A.A., Arevalo Ovalle, J.E., Madabhushi, A., González Osorio, F.A.: A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection. Medical Image MICCAI, Computing and Computer-Assisted Intervention (2013)

[24] Delgado, R., Tibau, X.A.: Why cohen's kappa should be avoided as performance measure in classification. PloS one 14 (2019)

[25] Devalla, S.K., Renukanand, P.K., Sreedhar, B.K., Subramanian, G., Zhang, L., Perera, S., Mari, J.M., Chin, K.S., Tun, T.A., Strouthidis, N.G., Aung, T., Thiéry, A.H., Girard, M.J.A.: Drunet: a dilated-residual u-net deep learning network to segment optic nerve head tissues in optical coherence tomography images. Biomedical optics express 9 pp. 3244–3326 (2018)

[26] Douglass, M.J.J., Géron, A.: Book review: Hands-on machine learning with scikit-learn, keras, and tensorflow: 2nd edition. Physical and Engineering Sciences in Medicine 43, 1135–1136 (2020)

[27] Drexler, W., Fujimoto, J.G.: Optical coherence tomography: Technology and applications. Biological and medical physics, biomedical engineering, Springer, Berlin (2008)

[28] Dreyfus, S.E.: Artificial neural networks, back propagation, and the kelley-bryson gradient procedure. Journal of guidance, control, and dynamics 13 pp. 926–928 (1990)

[29] Dubois, A., Levecq, O., Azimani, H., Siret, D., Barut, A., Suppa, M., Del Marmol, V., Malvehy, J., Cinotti, E., Rubegni, P., Perrot, J.L.: Line-field confocal optical coherence tomography for high-resolution noninvasive imaging of skin tumors. Journal of Biomedical Optics 23 pp. 1–9 (2018)

[30] Edgar, T., Manz, D.: Research methods for cyber security. Syngress (2017)

[31] Erb, R.J.: Introduction to backpropagation neural network computation. Pharmaceutical Research 10, 165–170 (1993)

[32] Fakoor, R., Ladhak, F., Nazi, A., Huber, M.: Using deep learning to enhance cancer diagnosis and classification. Proceedings of the 30th International Conference on Machine Learning, (2013)

[33] Fushiki, T.: Estimation of prediction error by using k-fold cross-validation. Stat Comput (Statistics and Computing) 21 pp. 137–146 (2011)

[34] Gao, K., Niu, S., Ji, Z., Wu, M., Chen, Q., Xu, R., Yuan, S., Fan, W., Chen, Y., Dong, J.: Double-branched and area-constraint fully convolutional networks for automated serous retinal detachment segmentation in sd-oct images. Computer Methods and Programs in Biomedicine 176, 69–80 (2019), `https://www.sciencedirect.com/science/article/pii/S0169260718318327`

[35] Gessert, N., Heyder, M., Latus, S., Lutz, M., Schlaefer, A.: Plaque classification in coronary arteries from ivoct images using convolutional neural networks and transfer learning. International Journal of Computer Assisted Radiology and Surgery 13, 1–273 (2018), `http://arxiv.org/pdf/1804.03904v1`

[36] Gessert, N., Lutz, M., Heyder, M., Latus, S., Leistner, D.M., Abdelwahed, Y.S., Schlaefer, A.: Automatic plaque detection in ivoct pullbacks using convolutional neural networks. IEEE Transactions on Medical Imaging 38, 426–434 (2019)

[37] Girard, M.J.A., Strouthidis, N.G., Desjardins, A., Mari, J.M., Ethier, C.R.: In vivo optic nerve head biomechanics: performance testing of a three-dimensional tracking algorithm. Journal of the Royal Society, Interface 10 (2013)

[38] Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. Proceedings of the Fourteenth Internal Conference on Artificial Intelligence and Statistics 15, 315–323 (2011), `https://proceedings.mlr.press/v15/glorot11a.html`

[39] Gonzalo, N., Tearney, G.J., Serruys, P.W., van Soest, G., Okamura, T., Garcia, H.M., van Geuns, R.J., van der Ent, M., Ligthart, J., Boum, B.E., et al.: Second-generation optical coherence tomography in clinical practice: High-speed data acquisition is highly reproducible in patients undergoing percutaneous coronary intervention. Revista Española de Cardiologia 63, 893–903 (2010)

[40] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, ACADEMIA, Accelerating the world's research. (2016), http://www.deeplearningbook.org

[41] Google Developers: Generalization: Definition, machine learning crash course (2022), https://developers.google.com/machine-learning/crash-course/generalization/video-lecture

[42] Grandini, M., Bagli, E., Visani, G.: Metrics for multi-class classification: an overview. arXiv preprint arXiv:2008.05756 (2020), http://arxiv.org/pdf/2008.05756v1

[43] Guha Roy, A., Conjeti, S., Carlier, S.G., Dutta, P.K., Kastrati, A., Laine, A.F., Navab, N., Katouzian, A., Sheet, D.: Lumen segmentation in intravascular optical coherence tomography using backscattering tracked and initialized random walks. IEEE journal of biomedical and health informatics 20, 606–614 (2016)

[44] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition pp. 770–778 (2015), http://arxiv.org/pdf/1512.03385v1

[45] He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. Computer Vision, ECCV pp. 630–645 (2016), http://arxiv.org/pdf/1603.05027v3

[46] Hee, M.R., Izatt, J.A., Swanson, E.A., Huang, D., Schuman, J.S., Lin, C.P., Puliafito, C.A., Fujimoto, J.G.: Optical coherence tomography of the human retina. Archives of Ophthalmology 113, 325–332 (1995)

[47] Heron, M.: Deaths: Leading causes for 2017: National vital statistics reports. National Vital Statistics Reports 68 (2017)

[48] Hiram G. Bezerra, Marco A. Costa, Giulio Guagliumi, Andrew M. Rollins, Daniel I. Simon: Intracoronary optical coherence tomography: A comprehensive review. JACC: Cardiovascular Interventions 2, 1035–1046 (2009)

[49] Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 4700–4708 (2016), http://arxiv.org/pdf/1608.06993v5

[50] Hussain, Z., Ginenez, F.: Differential data augmentation techniques for medical imaging classification tasks. AMIA Annu Symp Proc. (2018)

[51] Iqbal, H.: Plotneuralnet (2020), https://github.com/HarisIqbal88/PlotNeuralNet

[52] Jang, I.K., Bouma, B.E., Kang, D.H., Park, S.J., Park, S.W., Seung, K.B., Choi, K.B., Shishkov, M., Schlendorf, K., Pomerantsev, E., Houser, S.L., Aretz, H., Tearney, G.J.: Visualization of coronary atherosclerotic plaques in patients using optical coherence tomography: Comparison with intravascular ultrasound. Journal of the American College of Cardiology 39, 604–609 (2002)

[53] Kalkman, J.: Fourier-domain optical coherence tomography signal analysis and numerical modeling. International Journal of Optics 2017, 1–16 (2017)

[54] Kandel, I., Castelli, M.: The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. ICT Express 6 6, 312–315 (2020), `https://www.sciencedirect.com/science/article/pii/S2405959519303455`

[55] Kermani, A., Taki, A., Ayatollahi, A.: 3d analysis of thin-cap fibroatheromas by an automatic graph-based approach in intravascular optical coherence tomography. Archives of Cardiovascular Imaging Inpress (2016)

[56] Kihara, Y., Heeren, T.F.C., Lee, C.S., Wu, Y., Xiao, S., Tzaridis, S., Holz, F.G., Charbel Issa, P., Egan, C.A., Lee, A.Y.: Estimating retinal sensitivity using optical coherence tomography with deep-learning algorithms in macular telangiectasia type 2. JAMA network open 2 (2019)

[57] Kolluru, C., Prabhu, D., Gharaibeh, Y., Bezerra, H., Guagliumi, G., Wilson, D.: Deep neural networks for a-line-based plaque classification in coronary intravascular optical coherence tomography images. Journal of medical imaging (Bellingham, Wash.) 5, 044–504 (2018)

[58] Kostadinka Bizheva, Bingyao Tan, Benjamin MacLelan, Olivera Kralj, Mojtaba Hajialamdari, Denise Hileeto, Luigina Sorbara: Sub-micrometer axial resolution oct for in-vivo imaging of the cellular structure of healthy and keratoconic human corneas. Biomed. Opt. Express 8, 800–812 (2017)

[59] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Communications of the ACM 60(6), 84–90 (2017)

[60] Kumar Arvind, Sodhi Sartaj Singh: Comparative analysis of gaussian filter, median filter and denoise autoenocoder. 2020 7th International Conference 2020 pp. 45–51 (2020)

[61] Kuwayama, S., Ayatsuka, Y., Yanagisono, D., Uta, T., Usui, H., Kato, A., Takase, N., Ogura, Y., Yasukawa, T.: Automated detection of macular diseases by optical coherence tomography and artificial intelligence machine learning of optical coherence tomography images. Journal of ophthalmology 2019 (2019)

[62] Lang, A., Carass, A., Jedynak, B.M., Solomon, S.D., Calabresi, P.A., Prince, J.L.: Intensity inhomogeneity correction of sd-oct data using macular flatspace. Medical image analysis 43, 85–97 (2018)

[63] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. IEEE Conference (2015)

[64] Lu, H., Gargesha, M., Wang, Z., Chamie, D., Attizzani, G.F., Kanaya, T., Ray, S., Costa, M.A., Rollins, A.M., Bezerra, H.G., Wilson, D.L.: Automatic stent detection in intravascular oct images using bagged decision trees. Biomedical optics express 3, 2809–2824 (2012)

[65] Maas, A.L., Hannun, A.Y., Ng, A.Y., et al.: Rectifier nonlinearities improve neural network acoustic models. Proc. icml 30 (w2013)

[66] Mahmood, S.J.: Effects of Pre-Processed Training Data on Convolutional Neural Network's Training Accuracy where Training Dataset is Small. GRIN (2018), `https://www.grin.com/document/443889`

[67] Majib, M.S., Rahman, M.M., Sazzad, T.M.S., Khan, N.I., Dey, S.K.: Vgg-scnet: A vgg net-based deep learning framework for brain tumor detection on mri images. IEEE Access 9, 116942–116952 (2021)

[68] Neutelings, I.: Neural networks: Graphics with tikz in latex (2021), `https://tikz.net/neural_networks/#full_code`

[69] Nielsen, M.A.: Neural Networks and Deep Learning. Determination Press (2015), `http://neuralnetworksanddeeplearning.com/`

[70] Odaibo, S.G., D, M., S, M.: Generative adversarial networks synthesize realistic oct images of the retina, `http://arxiv.org/pdf/1902.06676v1`

[71] Oxford University Press.: Definition of heat map (2022), `https://www.lexico.com/en/definition/heat_map`

[72] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems 32 (2019)

[73] Pedregosa, F., Varoquaux, G., Gramford, A., Michel, V., Thirion, B.: Scikit-learn: Machine learning in python. Journal ofMachine Learning Research 12 (2011)

[74] PyTorch Contributors: Pytorch documentation: Pytorch is an optimized tensor library for deep learning using gpus and cpus. (2019), `https://pytorch.org/docs/stable/index.html`

[75] Reading, D.: tex-neural-network (2019), `https://github.com/dreading/tex-neural-network`

[76] Riebesell, J.: Random tikz collection: 107 standalone tikz images, mostly about physics and machine learning. (2022), `https://tikz.netlify.app`

[77] Righab, H., Ricardo, G.G., Said, G., Christophe, C.: Optical coherence tomography: From physical principles to clinical applications. Archives of Cardiovascular Diseases 105 pp. 529–534 (2012), `https://www.sciencedirect.com/science/article/pii/S1875213612001490`

[78] Ritchie, H., Roser, M.: Causes of death (2018), `https://ourworldindata.org/causes-of-death`

[79] Ruini, C., Schuh, S., Gust, C., Kendziora, B., Frommherz, L., French, L.E., Hartmann, D., Welzel, J., Sattler, E.: Line-field optical coherence tomography: in vivo diagnosis of basal cell carcinoma subtypes compared with histopathology. Clinical and experimental dermatology 46, 1471–1481 (2021)

[80] Sahoo, K.K., Dutta, I., Ijaz, M.F., Wozniak, M., Singh, P.K.: Tlefuzzynet: Fuzzy rank-based ensemble of transfer learning models for emotion recognition from human speeches. IEEE Access 9, 166518–166530 (2021)

[81] Schippling, S.: Optische kohärenztomografie (oct). Multiple Sklerose 2015 pp. 177–184 (2015)

[82] Shanker, M., Hu, M.Y., Hung, M.S.: Effect of data standardization on neural network training. Omega 24 pp. 385–397 (1996)

[83] Shipitko, O., Grigoryev, A.: Gaussian filtering for fpga based image processing with high-level synthesis tools. IITP RAS, Bolshoy Karetnyy per. 19 (2018)

[84] Simard, P.Y., Steinkraus, D., Platt, J.C., et al.: Best practices for convolutional neural networks applied to visual document analysis. Seventh International Conference on Document, Icda 3 (2003)

[85] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. ICLR (2015), `http://arxiv.org/pdf/1409.1556v6`

[86] Skaik, Y.A.E.W.: Understanding and using sensitivity, specificity and predictive values. Indian journal of ophthalmology 56 p. 341; author reply 341 (2008)

[87] Skillmon: K-fold cross-validation figure using tikz or table (2018), `https://tex.stackexchange.com/questions/429451/k-fold-cross-validation-figure-using-tikz-or-table`

[88] Srivastava, R.K., Greff, K., Schmidhuber, J.: Training very deep networks. Advances in Neural Information Processing Systems 28 (2015), `http://arxiv.org/pdf/1507.06228v2`

[89] Stifter, D.: Beyond biomedicine: a review of alternative applications and developments for optical coherence tomography. Applied Physics B 88 pp. 337–357 (2007)

[90] Tajbakhsh, N., Shin, J.Y., Gurudu, S.R., Hurst, R.T., Kendall, C.B., Gotway, M.B., Liang, J.: Convolutional neural networks for medical image analysis: Full training or fine tuning? IEEE Transactions on Medical Imaging 35, 1299–1312 (2016), `http://arxiv.org/pdf/1706.00712v1`

[91] Tanner, M.A., Wong, Wing Hung: The calculation of posterior distributions by data augmentation. Journal of the American Statistical Association 82, 528–540 (1987)

[92] Tearney G, Regar E, Akasaka T, et al.: Consensus standards for acquisition, measurement, and reporting of intravascular optical coherence tomography studies. Journal of the American College of Cardiology 59 pp. 1058–1072 (2012)

[93] Thimm, G., Fiesler, E.: Neural network initialization. From Natural to Artificial Neural Computation, IWANN pp. 535–542 (1995), `https://doi.org/10.1007/3-540-59497-3_220`

[94] Thoma, M.: Latex-examples (2015), `https://github.com/MartinThoma/LaTeX-examples`

[95] Torch Contributors: Models and pre-trained weights (2017), `https://pytorch.org/vision/stable/models.html`

[96] Tsantis, S., Kagadis, G.C., Katsanos, K., Karnabatidis, D., Bourantas, G., Nikiforidis, G.C.: Automatic vessel lumen segmentation and stent strut detection in intravascular optical coherence tomography. Medical physics 39 pp. 503–513 (2012)

[97] Twin, A.: Overfitting: What is overfitting? (2021), `https://www.investopedia.com/terms/o/overfitting.asp`

[98] Ughi, G.J., Adriaenssens, T., Sinnaeve, P., Desmet, W., D'hooge, J.: Automated tissue characterization of in vivo atherosclerotic plaques by intravascular optical coherence tomography images. Biomedical optics express 4 pp. 1014–1030 (2013)

[99] user121799: Plotting over/underfitting graph with tikz package (2022), `https://tex.stackexchange.com/questions/485900/plotting-over-underfitting-graph-with-tikz-package`

[100] user194703: visualizing matrix convolution (2019), `https://tex.stackexchange.com/questions/522118/visualizing-matrix-convolution`

[101] user30471: Replicating a plot using tikz (2020), `https://tex.stackexchange.com/questions/561921/replicating-a-plot-using-tikz`

[102] van Rijsbergen, C.J.: Information retrieval. 2nd. newton, ma (1979)

[103] Vani, S., Madhusudhana, R.T.V.: An experimental approach towards the performance assessment of various optimizers on convolutional neural network. Third International Conference on Trends in Electronics and Informatics (ICOEI) pp. 331–336 (2019)

[104] Welzel, J.: Optical coherence tomography in dermatology. Skin Research and Technology 7, 1–9 (2001)

[105] Wu, Z., Xu, G., Weinreb, R.N., Yu, M., Leung, C.K.S.: Optic nerve head deformation in glaucoma: A prospective analysis of optic nerve head surface and lamina cribrosa surface displacement. Ophthalmology 122 pp. 1317–1329 (2015), `https://www.sciencedirect.com/science/article/pii/S0161642015001797`

[106] Yabushita, H., Bouma, B.E., Houser, S.L., Aretz, H.T., Jang, I.K., Schlendorf, K.H., Kauffman, C.R., Shishkov, M., Kang, D.H., Halpern, E.F., Tearney, G.J.: Characterization of human atherosclerosis by optical coherence tomography. Circulation 106 pp. 1640–1645 (2002)

[107] Yerushalmy, J.: Statistical problems in assessing methods of medical diagnosis, with special reference to x-ray techniques. Public Health Reports (1896-1970) 62, 1432–1449 (1947), `http://www.jstor.org/stable/4586294`

[108] Yu, W., Yang, K., Bai, Y., Xiao, T., Yao, H., Rui, Y.: Visualizing and comparing alexnet and vgg using deconvolutional layers. 33rd International Conference on Machine Learning (2016)

[109] Zhao, W., Jenkins, M.W., Linderman, G.C., Bezerra, H.G., Fujino, Y., Costa, M.A., Wilson, D.L., Rollins, A.M.: 3-d stent detection in intravascular oct using a bayesian network and graph search. IEEE Transactions on Medical Imaging 34 pp. 1549–1561 (2015)

[110] Zhu, Q.: On the performance of matthews correlation coefficient (mcc) for imbalanced dataset. Pattern Recognition Letters 136, 71–80 (2020), `https://www.sciencedirect.com/science/article/pii/S016786552030115X`

[111] Zysk, A.M., Nguyen, F.T.O., Amy L., Daniel L. Marks, Stephen A. Boppart M.D.: Optical coherence tomography: a review of clinical development from bench to bedside. Journal of Biomedical Optics 12, 1–21 (2007)

Declaration

I hereby declare that the work in this thesis was composed and originated independently by myself and has not been submitted for another degree or diploma at any university or other institute. I certify that I have not used any sources or aids other than those indicated.

Date: May 30, 2022 ....
(Signature)