UNIVERSITÄT ZU LÜBECK

From the Institute of Mathematics and Image Computing
of the University of Lübeck
Director: Prof. Dr. rer. nat. Jan Modersitzki

# Deep Learning for Mass Spectrometry Imaging and Image Registration

Dissertation
for Fulfillment of
Requirements
for the Doctoral Degree
of the University of Lübeck

from the Department of Computer Sciences

Submitted by
Frederic Georg Kanter
from Münster

Lübeck, 2023

First referee: Prof. Dr. Jan Lellmann

Second referee: Prof. Dr. Mattias P. Heinrich

Date of oral examination: 30.06.2023

Approved for printing. Lübeck, 13.07.2023

# Abstract

In this thesis, we investigate new machine learning methods in the field of molecular diagnostics and image registration.

The first part is concerned with the analysis of mass-spectrometric data for pathological diagnostics. Mass spectrometry imaging methods are characterized by a particularly high resolution, which makes it possible to determine protein concentrations in individual tissue samples. This has great diagnostic potential for individualized medicine or the identification of new marker proteins. However, the efficient integration of mass spectrometry imaging into everyday clinical practice requires automated analysis. Neural networks are a suitable candidate for this purpose, and despite their numerous successes in other areas of medical imaging, they have hardly been used in the analysis of mass spectrometric data. Therefore, we address the application of neural networks for tumor classification based on spectrometric measurements. We develop a suitable architecture as well as methods for noise reduction, and compare the performance of our approach with the current state of the art. We apply our model to three datasets: the classification of four subtypes of ovarian cancer, the discrimination of amyloid plaques, and the discrimination of different types of pancreatic cancer.

The importance of image registration continues to grow with the steady increase in diagnostic imaging. In addition to classical variational techniques, neural networks are now widely used in this field. Both techniques have different strengths: variational approaches excel in accuracy, neural networks in speed. In the second part of this thesis, we present a new methodology to integrate a variational, iterative approach into a neural network. We aim to replace the updating rule, which is based on derivatives, with a nonlinear step computed by a network. We focus on large deformations and compare our model with classical variational methods.

# Zusammenfassung

Das übergreifende Thema dieser Arbeit besteht in der Entwicklung neuer Methoden des maschinellen Lernens für die molekulare Diagnostik und Bildregistrierung.

Der erste Teil behandelt die Analyse massenspektrometrischer Daten für die Diagnostik in der Pathologie. Massenspektrometrische bildgebende Verfahren zeichnen sich durch eine besonders hohe Auflösung aus, die es ermöglicht, Proteinkonzentrationen in einzelnen Gewebeproben zu bestimmen. Damit ist ein großes diagnostisches Potenzial für die individualisierte Medizin oder auch die Identifizierung neuer Markerproteine verbunden. Die effiziente Integration massenspektrometrischer Bildgebung in den klinischen Alltag erfordert jedoch automatisierte Analyseverfahren. Neuronale Netzwerke sind hierfür ein geeigneter Kandidat und werden trotz ihrer zahlreichen Erfolge in anderen Bereichen der medizinischen Bildverarbeitung bisher kaum in der Analyse massenspektrometrischer Daten eingesetzt. In dieser Arbeit werden deshalb neuronale Netzwerke zur Tumorklassifikation auf der Basis spektrometrischer Messungen entwickelt. Dazu gehören insbesondere die Entwicklung einer geeigneten Architektur, Methoden zur Entfernung von Rauschen und ein Vergleich der Leistungsfähigkeit der neuen Methoden mit dem aktuellen Stand der Technik. Das Modell wird anhand dreier Datensätze evaluiert: der Klassifikation von vier Ovarialkarzinom-Subtypen, der Unterscheidung von Amyloid-Plaques und der Unterscheidung unterschiedlicher Karzinomtypen im Pankreas.

Mit der stetigen Zunahme der bildgebenden Diagnostik nimmt auch die Bedeutung der Bildregistrierung weiter zu. Neben den klassischen variationellen Verfahren werden in der heutigen Zeit auch neuronale Netzwerke in diesem Bereich häufig eingesetzt. Beide Techniken haben unterschiedliche Stärken: Variationelle Ansätze sind in ihrer Genauigkeit noch ungeschlagen, neuronale Netzwerke dagegen in ihrer Schnelligkeit. Im zweiten Teil dieser Arbeit wird deshalb eine neue Methodik vorgestellt, um einen variationellen, iterativen Ansatz in ein neuronales Netzwerk zu integrieren. Im Mittelpunkt steht dabei der Ersatz der ableitungsbasierten Aktualisierungsregel durch einen nichtlinearen Schritt, der von einem Netzwerk berechnet wird. Insbesondere befasst sich die Arbeit mit großen Deformationen und einem Vergleich mit klassischen variationellen Verfahren.

# Acknowledgments

Many wonderful people have contributed to making this thesis possible. I want to thank them all for their support.

First of all, I would like to thank Jan Lellmann, who supervised this thesis. It is due to his support and tireless encouragement that this thesis could be written. I thank him for the many stimulating discussions, his constructive feedback and his precise criticism, from which I have learned so much.

Many thanks to all my former colleagues at the Institute of Mathematics and Image Computing (MIC) and Fraunhofer MEVIS. Each of you has contributed to making my time at the MIC one of great joy and fond memories. Special thanks go to Alessa and Sven, who shared an office with me and made my start at the institute so pleasant. I would also like to thank Annkristin and Danielle for taking the time to listen to my concerns on so many occasions. Many thanks to Max for his invaluable support in my work on learning-based registration.

Special thanks to Oliver and Herbert for their guidance and support throughout the MALDI project. They both answered countless questions about spectrometric imaging and made it possible for me to work with this technique.

Finally, I would like to thank my family and friends for their unconditional support and for being the source of so much needed energy. I especially thank Lisa for her great patience with me and the loving devotion with which she so often took care of me. Finally, I have to thank Leon for being the deadline that I had such a hard time setting for myself.

# Contents

# List of Peer-Reviewed Publications

The following is a list of the author's publications that were produced in the course of this thesis.

## Journal publications

[Kan+23] Frederic Kanter et al. "Classification of Pancreatic Ductal Adenocarcinoma Using MALDI Mass Spectrometry Imaging Combined with Neural Networks." In: *Cancers* 15.3 (2023).

[Kle+19] Oliver Klein et al. "MALDI-imaging for classification of epithelial ovarian cancer histotypes from a tissue microarray using machine learning methods." In: *Proteomics–Clinical Applications* 13.1 (2019), p. 1700181.

## Peer-reviewed conference proceedings

[KL22] Frederic Kanter and Jan Lellmann. "A Flexible Meta Learning Model for Image Registration." In: *International Conference on Medical Imaging with Deep Learning*. 2022, pp. 638–652.

# 1. Introduction

In modern clinical diagnostics, **biomedical imaging** is becoming increasingly important. It encompasses a number of different imaging modalities from light microscopy with very high spatial resolution to **tomography-based imaging** (CT, MRI). Tomographic imaging allows for non-invasive whole-body scans at the cost of relatively low spatial resolution. In contrast, **spectrometry-based imaging** such as Matrix-assisted laser desorption/ionization mass spectrometry imaging (MALDI-MSI) [Cha+06; Wal+08; Lei+09] and vibrational spectroscopy (Raman spectroscopy, infrared spectroscopy) yields detailed molecular information at high spatial resolution, but is limited to a confined region of the patient. Both branches are equally important for modern imaging-based medicine and pose interesting image processing challenges. One of these difficulties is the sheer volume of data that needs to be analyzed. Reliable, automated processing is often required to handle this amount of data efficiently. An important class of such methods is machine learning.

Recent methodical and technological advances in machine learning — most notably, **Convolutional Neural Networks (CNNs)** — have greatly enhanced the available toolset for analyzing biomedical imagery. The technical progress, such as GPU computing, enables faster computation times and a large increase of data sets, partly due to more frequent use of biomedical imaging in medicine, provides sufficient sample sizes for the data demanding training process [Lit+17; SWS17]. On the other hand, there have been important methodological advances that address common issues with large networks such as unsatisfactory convergence in training (stochastic and momentum-based optimization [KB14]), overfitting (dropout layers, weight sharing, CNNs), vanishing or exploding gradients (ReLU activation, normalization layers), and degradation (residual networks [He+16]). Since the work of Krizhevsky et al. [KSH12] achieved an impressive boost in accuracy at the ImageNet Challenge [Den+09] for image classification, convolutional networks provide state-of-the-art performance for a variety of imaging based tasks [Yan+15; Lee+09; Tho+18; SVL14; Hu+18].

Given the current success of machine learning in the field of medical imaging, in this thesis we propose approaches to two important problems: tumor classification using MALDI-MSI data and large deformations in image registration. In the first part, we use neural networks for the analysis of MALDI-MSI data, a technique that has been largely unexplored in this field. In the second part, we focus on the combination of classical iterative methods with neural networks in order to combine the advantages of both approaches.

## 1.1. Motivation

**MALDI-MSI:** Today, histochemical staining combined with light or electron microscopy is the most common method of analyzing tissue samples. A number of standard stains are used to visualize different cellular structures. For example, hematoxylin-eosin stains cell nuclei and cytoplasm. In addition, so-called histochemical stains allow the detection of active metabolism or individual metabolic products. This makes it easier for experts to evaluate tissue samples and enables various applications, such as the search for tumor-containing regions. However, this requires expert knowledge and there is often little to no automation. As a result, these methods are costly and can only handle small amounts of data. Other disadvantages include: no information on molecular structures, such as protein or lipid concentrations, and no way to correlate differentially expressed molecular profiles with tissue histology [Med+12]. However, the histological analysis of tissue samples is an integral part of modern diagnostics but is limited by its time-consuming and labor-intensive processing steps.

In contrast to staining and microscopy techniques, spectrometric imaging, in particular MALDI-MSI, allows precise histological classification or identification of biomarkers and is an essential tool in advanced molecular diagnostics and personalized medicine. Besides its advantages, mass spectrometry imaging produces complex, high-dimensional data, which makes analysis and interpretation computationally challenging. Currently, the data size limits the clinical impact of this promising diagnostic tool. In **mass spectrometric imaging** (Figure 1.1), the data allows us to draw conclusions about the spatial molecular composition of the tissue and holds great potential for applications in pathology, diagnosis, analysis of tumor growth, outcome prediction, and analysis of drug uptake. MALDI-MSI typically generates 3D datasets with $>10^5$ channels per voxel and thus remains challenging to process adequately, limiting its deployment in histological routine.

Nevertheless, to enable the use of this technology, the size of the resulting data must be reduced. Traditionally, such high-dimensional data is first processed using dimension-reduction techniques such as peak detection and (non-)linear PCA, with subsequent analysis using classical machine learning tools such as random forest classifiers, support vector machines, and hierarchical clustering [Gal+16]. We believe that neural network-based machine learning provides an alternative to those classical methods.

After gaining popularity under the label of deep learning [LBH15], CNNs have been extended to generative models [Goo+14], and for augmenting classical optimization methods [AÖ17]. A new machine learning ecosystem of tools (TensorFlow [Mar+15], PyTorch [Pas+19]) and pre-trained models reduce the computational effort and provide better accessibility.

However, applying modern machine learning methods to **data with a very large number of channels**, such as diffusion-weighted MRI and spectroscopy imagery has been much less studied [Beh+17], and is therefore a promising research direction. Additionally, in the imaging and computer vision communities, so far most effort in developing CNN-based approaches has gone into processing data with two, less commonly three, spatial dimensions and usually one to three channels, such as intensity values or RGB data.

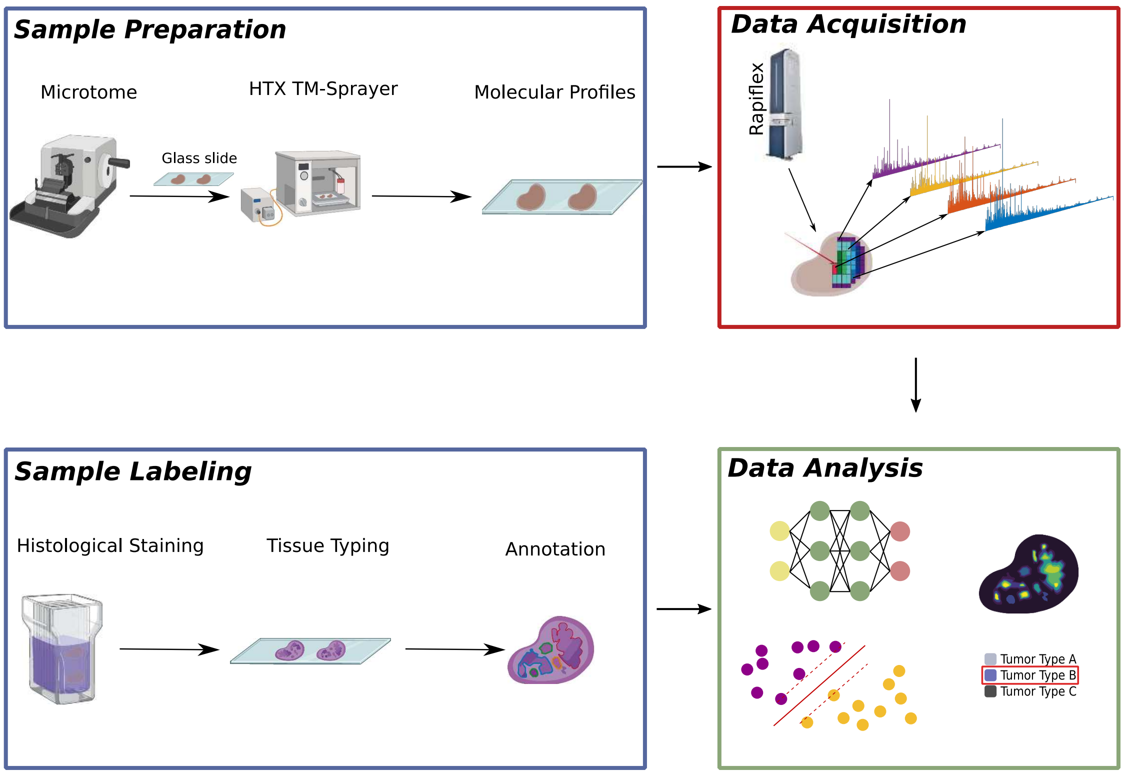


Figure 1.1.: Process pipeline of MALDI-MSI combined with machine learning. **Top left**: Preparation of tissue sample before data acquisition. The tissue taken from the patient is washed with alcohol, dried, and positioned on a glass slide. A chemical matrix is applied to provide the environment for laser absorption and correct charge. **Top right**: Rapiflex tissue typer acquires the spectral data. A laser extracts molecules from the tissue, which have single charge and are accelerated towards a time-of-flight detector by a constant magnetic field. Each detection time correlates to a characteristic molecule or isotope thereof. **Bottom left**: A pathological expert uses chemical staining to analyze the tissue and marks tumor regions by attributing class labels. **Bottom right**: The label and spectral data are then merged to create the data set. The high resolution of MALDI-MSI allows particularly accurate differentiation based on different molecular structures. Combined with automated classification procedures, tissue samples can be analyzed quickly and efficiently. This automated processing of large amounts of data is beneficial for pathological diagnostics. Typical existing classification algorithms are of linear nature; in this thesis we propose non-linear neural networks as a replacement.

Such data has a strong spatial coherence in the two or three spatial dimensions, but no useful notion of spatiality in the channel dimension. This work aims to fill this gap and apply neural network architectures to high-dimensional multichannel spectrometry imaging data.

To make routine use of MALDI-MSI in diagnostics even more attractive, we are working on automated analysis using neural networks to enable rapid processing of large cohorts of tissue samples. In this work we provide classification results for three real-world use cases:

**Epithelial ovarian cancer** (EOC) is an inhomogeneous disease with multiple histological subtypes. Five main subtypes have been described by the World Health Organization with diverse molecular structures, clinical behavior, and therapeutic prognoses [Mei+16; PDE18]. Response to standard therapy can differ vastly and clinical outcomes improve with adapted therapy approaches for each EOC subtype.

The correct histotyping of ovarian carcinomas is important, as the diverse molecular features of various subtypes require different approaches in clinical management, including chemotherapeutic strategies and targeted therapies. In Chapter 3 (Section 3.5), we focus on four distinct subtypes Serous Ovarian cancer (OC), high-grade serous ovarian cancer (HGSOC), low-grade serous ovarian cancer (LGSOC), and serous borderline tumors (sBOT). The classification is particularly challenging due to areas with morphological overlaps of multiple subtypes.

In this thesis, we aim to provide learning-based algorithms to distinguish between these ovarian subtypes on such a precise level. These methods allow the full potential of MALDI-MSI to be unleashed and provide an automated alternative to methods currently used in routine clinical practice such as histology and extensive immunohistochemistry staining. These approaches rely on the detection of novel biomarkers, which are still sparse in the routine. As a result, the misdiagnosis rate is around 15% [Köb+14] and there is a clear medical need to define new molecular classifiers.

As a solution, we propose learning-based classifiers implemented by various neuronal networks. In Chapter 3 we consider architectures with and without convolutions, experiment on possible feature extraction methods, and compare our methods with non-learning based approaches.

**Amyloid** fibrils are abnormal proteins that can occur in a multitude of organs such as the heart, kidneys, nervous system, musculoskeletal system, or gastrointestinal organs [Ben10]. About 15 million amyloidosis disease cases occur per year if the AL and AA subtypes are counted as one [Haz13]. The annual death rate is 1 in 1.000 people for all systemic amyloidosis types [MB03] and without treatment the life expectancy of patients can drop to six months [Haz13].

A variety of different amyloidosis types exist, distinct from another by type-specific protein misfoldings [And+13]. The most common types of amyloidosis are light chain (AL), inflammation (AA), dialysis-related ($A\beta_2M$), and hereditary (ATTR) [Haz13]. The symptoms of amyloidosis disease are manifold and also vary greatly between affected organs [Ger+15]. Since treatment focuses on decreasing the concentration of the causative protein, distinction between the subtypes is of utmost importance and diagnosis can only be confirmed reliable by tissue biopsy [Haz13].

In the clinical workflow, microscopes, staining, and immunohistochemistry are commonly used to determine the amyloidosis subtype. However, the immunohistochemistry method tends to miss AL classification [EN08]. By contrast, mass spectrometry is the most reliable method for identifying the different forms of amyloidosis [RL11]. Therefore, we aim to combine MALDI-MSI and machine learning methods to reliably distinguish between the types ATTR and AL subtypes, which are difficult to separate but benefit from customized treatments (Section 3.6).

Our learning based methods provide fast, reliable classification with potential speed up for diagnosis, due to replacing the time-consuming manual classification with an automated approach. To this end, we extend the methods from Section 3.5 to include recurrent and residual networks that take more account of the structure of MALDI-MSI spectra.

**Pancreatic ductal adenocarcinoma** (PDAC) is the most common neoplastic disease of the pancreas and accounts for more than 90% of all pancreatic malignancies and with poor prognosis [Kle+16]. With a 5-year overall survival of less than 8% PDAC is the fourth most frequent cause of cancer-related deaths worldwide [Bra+18]. Projections indicate that the number of PDAC diagnoses as well as PDAC-related deaths will more than double in the next decade in the United States [Rah+14] and in European countries [Qua+16]. Lifestyle habits, such as alcohol and tobacco abuse, are known to increase the risk of and appear to be involved in the development of PDAC [Gap+11; Pel+14; Ols+10; Del+16]. Due to the increase in PDAC cases in the future, accurate and reliable diagnostic is essential but currently not available.

In contrast to immunohistochemistry staining, the spatially distinct signatures of MALDI-MSI spectra can be obtained in high throughput in a clinically feasible time frame. In addition, the method is more cost-effective and could provide a new dimension to the current classification of patient subgroups, potentially assisting the prediction of disease progression or resistance development. Our methods are designed to automate the analysis of MALDI-MSI data, enabling reliable and rapid diagnosis of PADC.

Differentiating PDAC from other pancreatic cancers is complicated. We therefore extend our methodology with the Transformer architecture, which has shown great success in processing sequential data similar to MALDI-MSI spectra. We also propose a new filtering method to address the problem of uninformative spectra (Section 3.7).

**Image Registration:** The process of finding spatial correspondences between two or more images is called *image registration* (Figure 1.2). It is a core task in computer vision and medical image analysis, where the alignment of images for comparison is of particular interest. Accompanying increasing availability of tomographic imaging, the importance of image registration in clinical diagnostics as well as the amount of image data have grown in recent years. It is a crucial tool for improving imaging-based diagnostics such as motion correction, intra-operative fusion of different modalities, or change detection in follow-up studies [SDP13; Fat+15; Lan+09; HDI+11]. Various algorithms have been established for solving image registration problems [Thi98; Rue+99; Ami94]. In this thesis we adopt a broad categorization and focus on two popular classes: classical

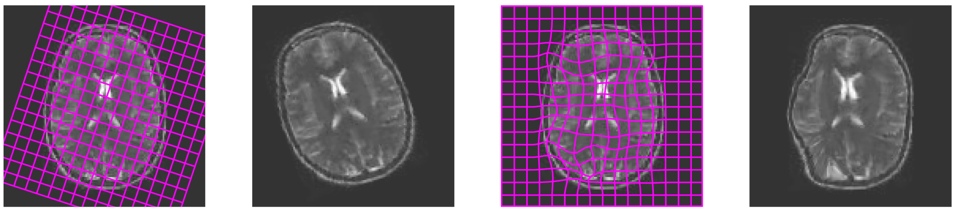


Figure 1.2.: In image registration the deformation needed to map two images onto each other can be defined on *regular grid* (magenta). **Right**: Example affine deformation with $n = 6$ degrees of freedom (DOF), consisting of rotation, translation, and shearing of the object. Affine deformations act globally, transforming all image points in the same manner. The grid on the right side visualizes the transformation yielding the underlying brain with its starting position as on the left side. **Left**: Example *deformable* deformation where the DOF for the transformation is typically proportional to the number of image points. Deformable transformations are characterized by local displacement, clearly visible on the left side of the brain. The transformation of image points can vary in every point, with some areas containing large movements and other areas with little to no movement.

variational methods [Mod03; Ami94] and neural network-based approaches [Bal+19; She+19; MC21b; BHH21].

Current variational image registration models are very accurate, but also complex and — especially for large images — computationally expensive. They often exhibit a long computation time and may require large amounts of memory. Naturally, clinical requirements to image registration include short processing times. There is a potentially large benefit of accurate registration algorithms in diagnosis and treatment [RM03]. Low run-times could also enable new forms of supporting clinicians in real-time scenarios. Recent studies either worked on acceleration through improved hardware use [Sha+10], elaborated computation schemes [Kön+18], or omitting the variational approach in favor of deep learning algorithms [Bal+19; HGH19; She+19; BBH20].

In contrast to variational methods, neural networks learn to solve tasks in a problem-adapted and data-driven manner using training examples. Neural network approaches rely on suitable training data, which is scarce in the medical context. Architectures and optimization strategies can vary strongly [HKY20] and models are often highly specialized on a given task or image modality [HGH19]. A popular strategy to boost performance is the use of auxiliary losses [Her+19], adding additional energy terms to the distance-based loss formulation and consequently increasing model complexity.

Despite their undisputed success in many areas of machine vision, neural networks do not yet achieve the level of accuracy of classical methods that have been researched and optimized in the context of medical image registration. In addition, neural networks appear to struggle to combine corrections for both large global and small local deformations and existing models typically resort to predicting small non-linear deformations after a classical pre-registration.

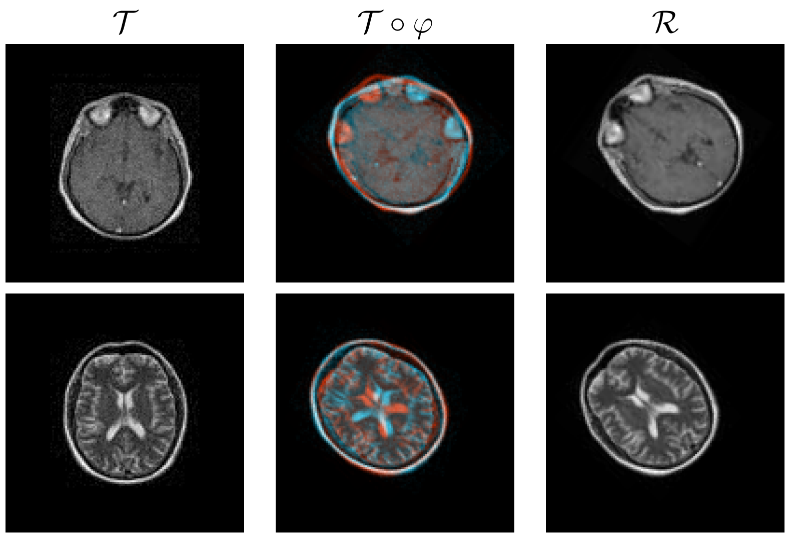


Figure 1.3.: Examples for failed registration by iterative methods. Here a multi-level scheme and an L-BFGS optimizer are used. The implementation is taken from the *elastix* toolbox. Even for affine transformations $\varphi$, local iterative methods can get stuck in local minima, and the line search can fail. The **left** and **right** side show the template image $\mathcal{T}$ and reference image $\mathcal{R}$. **Center**: Application of the computed transformation $\mathcal{T} \circ \varphi$ yields an incorrect registration result. In this case, the object is characterized by a nearly-symmetric circular shape causing the distance function to be highly non-convex with multiple local minima.

Both methods exhibit noticeable drawbacks. For variational methods, robustness of the optimization is a large concern. These methods depend on good initial estimates and well-tuned hyperparameters. In a highly non-convex setting they tend to get stuck in local minima (Figure 1.3).

Neural networks provide fast inference, but lack accuracy and rely more heavily on data. They are not as effortlessly applicable to new registration tasks, as they commonly require to be refined on the new data set, and do not generalize in the same way as classical approaches.

Since both approaches have opposite strengths and weaknesses, our goal is to combine classical variational methods and neural networks. In this way, each approach can benefit from the strengths of the other. This constitutes the motivation of the second part of this thesis. We propose a trainable architecture for image registration to produce robust starting points for classical image registration methods. We employ a combined method to leverage the advantages of classical iterative and learning-based methods (Chapter 5).

In related work, neural networks are used to predict the initial momentum of LDDMM [Yan+17], as a regularization tool in a conventional registration model [NKV19], or to create high-level features for subsequent registration [BH19].

## 1.2. Contributions and Outline

The main contribution of this thesis is two-fold. First, we provide automatic learning-based classification algorithms for high-resolution histological tissue measurements. The processed data is of high complexity, since histological tissue typing suffers from morphological similarities between examined classes and the experimental studies contain noisy measurements. In addition, class labels are applied to tissue sections in whole and measurements within do not necessarily contain pathological tissue. Secondly, we propose a method that bridges the gap between variational and learning-based image registration in order to emphasize that combining both established approaches is a promising field of study.

**Deep Learning for MALDI-MSI.** In the first part, we develop novel *neural network classifiers* for MALDI-MSI. Prior work on neural networks in combination with MALDI-MSI data is sparse, therefore we focus on identifying a suitable architecture. We analyze the influence of established pre-processing steps and design additional pre-processing. We derive a novel process pipeline for complex MALDI-MSI data composed of

- a novel preprocessing *filter*, followed by
- a (C-)NN-based network with several possible *architectures*.

All models are analyzed and different parameter settings are discussed. We compare our methods to established methods commonly used for automatic tumor classification in MALDI-MSI. Moreover, we test our methods on three clinical applications: ovarian cancer, amyloidosis, and pancreas cancer. All three tasks are of great clinical relevance, diagnosis is currently performed without automatization, and costly expert supervision is needed. Our methods have been published in [Kle+19; Kan+23] for the ovarian and pancreas cancer data sets and are extended in this thesis by the amyloidosis data set. Compared to our publications [Kle+19; Kan+23], we provide a more comprehensive analysis and expand on the insights of preprocessing as well as the influence of noise in MALDI-MSI data.

**Meta-Learning for Image Registration.** In the second part, we focus on the fusion of classical and neural network-based image registration. A novel network model is introduced using strategies from meta learning and incorporating successful classical optimization strategies, such as a multi-level scheme and using past steps in the update calculation of an iterative scheme. Our model is characterized by

- a shallow architecture utilizing a *hidden state approach*,
- an *iterative update* scheme, and
- *non-linear updates*.

A major section of this part is dedicated to the design of an LSTM-based network for iterative image registration. We evaluate the performance of the plain network and when combined with the established L-BFGS optimizer. Both variants are compared to similar methods implemented in the *FAIR* and *elastix* toolboxes. For affine deformations, we have published the approach in [KL22]. In this thesis, we additionally extend the method to non-parametric deformations, a more common form in medical image analysis. All models are evaluated on a synthetic x-ray image data set, an MRI brain data set from the *kaggle* platform, and the publicly available *fastMRI NYU* data set.

**Outline.** Following this two-part setup, this thesis is structured as follows. In Chapter 2, a general introduction to deep learning is given in order to provide a foundation for all models throughout this thesis. Moreover, the concept of meta learning is discussed briefly as a basis for the methods in Chapter 5. For each type of architecture, related work as well as a brief overview of applications are provided.

In Chapter 3, the introduced network architectures are applied to MALDI-MSI data. As the combination of neural networks and MALDI-MSI data is a novel field of research, suitable models are first designed using the layer structures from Chapter 2. Different preprocessing steps are evaluated and the influence on the performance is analyzed. The best-performing model is then compared to widely-used methods including linear discriminant analysis (LDA) and support vector machines (SVM). The performance of all methods is evaluated on three clinical use cases of tissue-typing.

The second part of this thesis starts with Chapter 4, where a general introduction to image registration is given. The variational, iterative model serves as the basis for this thesis. Important details concerning discretization, numerical optimization, and advanced strategies such as the multi-level scheme are discussed. In addition, a brief introduction to the used benchmark toolboxes *FAIR* and *elastix* is provided.

The novel design for the registration network is laid out in Chapter 5. Training routines, data creation, and scheduling techniques are discussed in more detail, owing to the more important role in comparison to part one of this thesis. The model is benchmarked against methods from the established registration toolboxes *FAIR* and *elastix*.

This thesis concludes with a summary and discussion in Chapter 6, where we position our work in the context of machine learning in MALDI-MSI and image registration and provide an outlook on both methods.

# 2. Deep Learning Fundamentals

We begin with an introduction to neural networks (NN), a method of machine learning. First, key features of these methods are summarized in Section 2.1. These build the basis of all methods presented in this thesis. Different types of network architectures are used to solve problems in the fields of image classification and registration.

In Section 2.2, sequence models are characterized and in Section 2.3 Transformer architectures are introduced. An overview of their properties is given and differences to the more common neural network architectures are discussed. As the second part of this thesis focuses on applications of meta-learning strategies in the context of image registration, the relevant concepts are summarized in Section 2.4.

## 2.1. Neural Networks

An artificial neural method can be viewed as a way of describing a function $f$ mapping an input vector to an output vector by means of a collection of nonlinearities (**notes**) connected by weights (Figure 2.1). These weights are typically learned in order to approximate a certain — only partially known — function $\hat{\mathbf{f}}$. In the following we will use a representation based on alternating linear and nonlinear layers, which grasp the action of the weighted connections and nonlinearities:

$$\mathbf{f} = f_N(...f_2(f_1(x, \theta_1), \theta_2), \theta_N), \quad N \in \mathbb{N}, \tag{2.1}$$

where $x$ is the input and can represent diverse data, such as images or audio signals. For a classification task, the neural network maps the input $x$ to a category $y$

$$x \mapsto y \coloneqq \mathbf{f}(x, \Theta) \tag{2.2}$$

with $\Theta$ designating the trainable parameters of $\mathbf{f}$. The functions $f_i$ are often called **layers** and are the main building block of each neural network. Each layer consists of a variable number of $j$ units, also named neurons regarding their biological equivalents in the human brain. A layer can be described as

$$f_i(x, \theta_i) = \sigma_i \left( \sum_j \theta_{i,j} x_j - b_i \right). \tag{2.3}$$

$\theta_{i,j}$ is the weight of the layer, $f_i$ at the position $j$, $b_i$ is the bias, and $x$ is the input. Together, the weight $\theta_{i,j}$ and biases $b_i$ form the parameter vector $\Theta$. Here $\sigma_i$ denotes an often nonlinear activation function. Neural networks with many layers are named

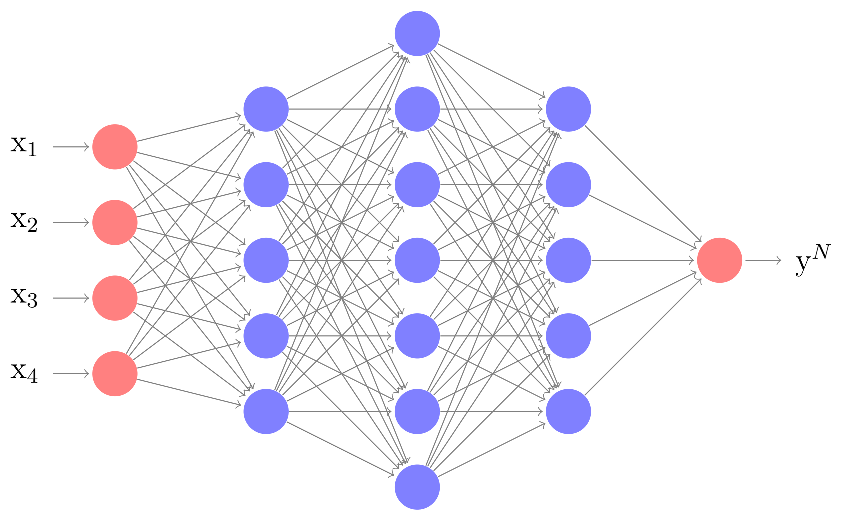


Figure 2.1.: Sketch of a neural network with $N = 5$. The red layers $f_1$ and $f_5$ are the input and output layers and are defined by the size of the input $x$ and output $y^N$. Here a single unit is displayed, which can be used for two-fold classification or regression tasks. The blue layers ($f_2$-$f_4$) are referred to as **hidden layers** and are variable in terms of the number of units per layer. Grey edges are associated with trainable weights that are iteratively adopted in order to approximate the desired input-output mapping [Kan18].

**Deep Neural Networks** (DNN) and have been highly successful in many computer vision tasks such as image classification [HSS18; SZ15], object detection [He+17; CV18], and image registration [Bal+19; HGH19; Hei19]. The activation function $\sigma_i$ defines the nonlinear part of $f_i$, while the multiplication of the parameter weights $\theta$ with the input $x$ is a linear - more precisely, affine - operation. The choice of $\sigma$ is a crucial part of designing neural networks, therefore the most frequently used are described in the next section.

### 2.1.1. Activation Functions

The nonlinear component has multiple tasks in the context of neural networks. Firstly, it enables the approximation of nonlinear objectives. Secondly, it stabilizes the optimization process used to train the network. Classical activation functions, in particular the sigmoid function, map the output of the weight input multiplication to a finite interval

$$\text{sig}(z) = \frac{1}{1 + e^{-z}}. \tag{2.4}$$

In addition to the sigmoid function, the hyperbolic tangent

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = 1 - \frac{2}{e^{2z} + 1} \tag{2.5}$$

is also used in this thesis. Both are integral parts of the LSTM architecture, which is introduced later on in Section 2.2 and is an important component for our model in

Chapter 5. From (2.4) and (2.5) it follows that both functions converge to 1 for large values. For small values, on the other hand, the sigmoid function converges to 0 and the hyperbolic tangent to -1. For this reason, many different inputs are mapped to similar outputs in the margins.

This property of **activation** functions tend to hinder the optimization process due to an effect called vanishing/exploding gradient: Close to the interval boundaries even large weight updates lead to very small — or no — changes in the output. This situation often occurs in very deep networks with a large number of layers. The most prominent activation function to tackle this problem is the **Rectified Linear Unit** (ReLU), defined as

$$\mathrm{ReLU}(z) = \max(0, z). \tag{2.6}$$

In recent years improvements in the form of leaky ReLU and others further, diminish the occurrence of the phenomenon for values below 0. There are many different variants of activation functions, some are designed for a special purpose others have shown favorable properties for many use cases. In this thesis, the effects of varying activation functions are not investigated and the ReLU activation is used for most layers except within the aforementioned LSTM layer.

For our classification models in Chapter 3 we also employ the softmax activation

$$\mathrm{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}. \tag{2.7}$$

The activation returns pseudo-probabilities, which then serve as a basis to classify the data.

### 2.1.2. Layer Architectures

We have established the most basic building blocks of each neural network. Following this introduction, we will discuss some layer architectures, which define the form and properties of each $f_i$.

All layers depicted in Figure 2.1 are fully connected layers; each unit in the layer has a connection to each unit in the following layer. The input of each unit in the second layer consists of the outputs of every unit in the first layer. The relation between these two layers can be formulated as a matrix-vector multiplication with a dense matrix $\Theta_i$:

$$z_{i+1} = \sigma_i(\Theta_i z_i - b_i), \tag{2.8}$$

where $\Theta_i$, $b_i$, $z_i$ are the weights, biases, and outputs of layer $f_i$, and $\sigma_i$ is an activation function.

Networks based on fully connected layers can solve many complex problems but do not exploit any topological structures of the input data, such as spatial or temporal neighborhood relationships. Furthermore, these networks are memory-demanding due to the high parameter count. A highly successful alternative is based on the mathematical convolution operation:

$$z_{i+1} = \sigma_i(\Theta_i * z_i - b_i), \tag{2.9}$$

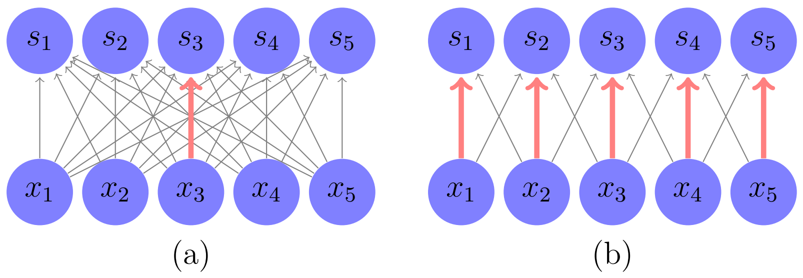


Figure 2.2.: Weight sharing. The red arrows denote identical (shared) weights (a) fully connected model, (b) 3×3 kernel in a convolutional model. Weight sharing allows sacrificing some generality, which reduces the parameter count, while preserving a reasonable amount of expressiveness, and allows to guarantee certain (spatial and temporal) invariances [Kan18].

where $*$ is a (typically 1D to 3D) convolution, applied on one or more dimensions of the input activations. These kinds of networks, therefore, are named **Convolutional Neural Networks** (CNN) and have shown tremendous success in various computer vision tasks [KSH12]. Figure 2.2 shows the links between layers for fully connected (left) and convolutional (right) networks.

Another important layer type is the **pooling** layer. In contrast to the aforementioned layer forms it does not produce input activations through matrix multiplications but aggregates neighboring activations. Pooling assists to stabilize the network performance. It further increases invariance to small translations and improves computational efficiency — by reducing the dimensions by a constant factor —, and allows to generate networks that operate on a hierarchy of resolutions. Although pooling has beneficial effects for machine learning purposes, it results in loss of information due to its reductive nature [GBC16]. Frequently used pooling operations are the maximum and average operations applied to a small neighborhood, following a fully connected or convolutional layer and the associated nonlinearity.

**Residual connections** are skip-connections between layers in a neural network. In their work [He+16] the authors found that networks with more layers exhibit worse accuracy than similar networks with fewer layers. This was not only due to the vanishing gradient problem, which the authors accounted for by using normalization before and after gradient calculation. They concluded that deeper networks are inherently more difficult to optimize. As a solution, they propose "shortcut functions" to formulate the mapping in (2.2) as a residual function

$$y = \mathbf{f}(x, \Theta) + x. \tag{2.10}$$

The shortcut term — adding $x$ — neither introduces additional parameters nor adds computational complexity. The new model is able to learn the perturbations with reference to the identity function instead of learning a new mapping for each layer. The authors provide experimental results that learned residual functions produce smaller gradients and thus provide reasonable preconditioning.

Networks with residual connections outperform their plain counterparts on the classification tasks ImageNet [Den+09] and CIFAR 10 [Kri09]. The authors in [He+16] argue that the residual connections facilitate training in the first epochs for very deep networks. For this purpose, they consider the standard deviations of the layer responses before passing them into the nonlinear activation function. For the networks with residual connections, the analysis reveals smaller responses than their plain counterparts. Also, the responses' magnitudes are lower and when multiple layers are used, this causes the single layer to modify the signal less. We make use of residual connections in our model architectures in Chapter 3.

Most neural networks are composed of these layer building blocks. In general, the forward pass — defined in (2.2) — through one layer can be summed up as follows. The input data is processed by the layer either by matrix multiplication or convolution using a linear operator $\mathbf{\Theta}_i$, here $i \in [1, N]$ is the layer index, and some form of pooling may be applied. Finally, the output is transformed by the activation function in a nonlinear fashion. This whole pipeline is highly flexible and further operations can be added to it. We will discuss some of these extensions later in Section 2.2 and Section 2.3.

### 2.1.3. Training

Now that we know the different types of layers, the question arises as to what is necessary to make neural networks solve new tasks. We have already seen that the mapping function $\mathbf{f}$ is parameterized by the weights of the layers and that these are adapted by learning on samples of the tasks to be solved. In order to find suitable values for the weights $\Theta$ so that the networks successfully solve the tasks, an optimization — or, *training* — process is required.

We can cast the approximation of the $\widehat{\mathbf{f}}$ by $\mathbf{f}$ as an optimization problem using an error metric, for example, the mean squared error (MSE)

$$L(\theta) = ||\frac{1}{2} \sum_{x \in \mathbb{R}^m} (\widehat{\mathbf{f}}(x) - \mathbf{f}(x; \theta))||_2^2. \tag{2.11}$$

Here the input vector $x$ has the dimension $m$. In the machine learning context, $L$ is referred to as **cost** or **loss function**. Given the complex structure of the loss, typically a large number of examples, and high dimensionality of $\Theta$, currently the most viable option is to find a minimizer of the problem

$$\min_{\theta \in \mathbb{R}^N} \; L(\theta) \tag{2.12}$$

using an iterative first-order method such as **Gradient Descent** (GD). Gradient descent adjusts $\theta$ step-wise along the opposite direction of the function gradients $\nabla_\theta L$:

$$\theta_{t+1} = \theta_t - \epsilon \nabla_\theta L \quad t \in [1, T]. \tag{2.13}$$

The *learning rate* $\epsilon$ controls the update size. An effective way to derive the gradient involves the **back-propagation algorithm** proposed by Rumelhart [RHW86]. The forward pass through a neural network $\mathbf{f}$ is defined in (2.2), an input $x$ is mapped to an output $y$. In the end, the forward pass produces a scalar cost $L(\theta)$, an example is given in (2.11). The back-propagation algorithm enables the information gained in the cost function to flow backward through the network to compute the gradient $\nabla_\theta L(\theta)$. One important ingredient in back-propagation is the chain rule of calculus.

In the most general sense, the full network including the loss function can be viewed as a composition of functions,

$$L(\theta) = f_3(f_2(f_1(\theta))), \tag{2.14}$$

where the $f_i$ depends on the input data $x$; in this simple example, the network has three layers. In order to minimize the loss, we need to compute the gradient of $L(\theta)$ with respect to the weights $\theta$. This can be achieved by efficiently evaluating the chain rule:

$$\nabla L(\theta) = \nabla f_1(\theta) \cdot (\nabla f_2(f_1(\theta)) \cdot (\nabla f_3(f_2(f_1(\theta))))), \tag{2.15}$$

where $\nabla$ denoted the generalized gradient (transposed Jacobian). The back-propagation algorithm performs these Jacobian-gradient products for all layers in a network. In the general case, the nonlinear nature of neural networks causes $L$ to be non-convex. Therefore, gradient descent, which uses local updates, cannot be expected to converge to a global solution. In the field of deep learning an adapted form of the gradient descent, the **Stochastic Gradient Descent** (SGD) is used.

The generalization of neural networks demands large data sets, which makes evaluating $\nabla L$ computationally expensive. The cost function $L$ can be seen as per-sample cost functions $L_m$ summed over all samples in the data set. The computational cost can be reduced by computing $L(\theta)$ and $\nabla_\theta L$ over a subset of $m$ samples randomly drawn from the data set [GBC16]. This subset or *minibatch* results in a new *batched cost function*

$$L_m(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (\widehat{\mathbf{f}}(x^{(i)}) - \mathbf{f}(x^{(i)}; \theta))^2. \tag{2.16}$$

The full gradient $\nabla_\theta L$ can be viewed as an expectation taken over the samples $x^{(i)}$. In SGD, it is estimated by

$$g_t = \nabla_\theta L_m \tag{2.17}$$

using the $m$ samples from the mini-batch, where $m$ is considerably smaller than the total number of samples in the data set. We denote $g_t$ as the gradient of $L(\theta)$ for step $t$. The update step in SGD takes the form of

$$\theta_{t+1} = \theta_t - \epsilon g_t, \tag{2.18}$$

where $\epsilon$ is the step size or *learning rate*. Although the SGD method does not ensure convergence for non-convex functions, it enables a quick and sufficient reduction of the cost function in practice [GBC16].

**Advanced Optimizers.** In this thesis, the **Adam** optimizer is used instead of plain SGD, as introduced by Kingma et al. in 2014 [KB14]. It is one of the most commonly used optimizers to date, due to its good convergence characteristics [Déf+22] and stability against noisy or sparse gradients. The Adam optimizer utilizes adaptive estimates of lower-order moments to compute the weight updates. It incorporates an adaptive learning rate approach based on previous gradient steps and is an extension of two other adaptive learning rate methods AdaGrad [DHS11] and RMSProp [Hin12].

Starting from batch-wise gradient calculation in (2.17), we keep the strategy of gradient calculation on subsets. The novelty of Adam is the introduction of moving averages of the gradient

$$\begin{aligned} m_t &= \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \\ \hat{m}_t &= m_t/(1-\beta_1^t) \end{aligned} \tag{2.19}$$

and the element-wise squared gradient

$$\begin{aligned} v_t &= \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \\ \hat{v}_t &= v_t/(1-\beta_2^t) \end{aligned} \tag{2.20}$$

with their respective decay controlling parameters $\beta_1, \beta_2 \in [0, 1)$. The averages $m_t$, $v_t$ estimate the first moment (2.19) and second moment (2.20) of the gradient. Due to the initialization of both averages with 0, the moments are biased towards zero. For this reason, the authors suggest counteracting this effect by scaling the averages with a factor $1/(1-\beta_i^t),\ i \in \{1, 2\}$. Adding these changes to the weights update in (2.18) yields

$$\theta_t = \theta_{t-1} - \epsilon \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \zeta). \tag{2.21}$$

The recommended default setting for the stabilization parameter $\zeta$ is $10^{-8}$. In contrast to SGD the update for $\theta$ is not depending directly on $\nabla_\theta$ but on two moment estimates $m_t$ and $v_t$ derived from the gradient. As a result, Adam shows better convergence rates and established itself as the default optimizer in a neural network learning setting. For a more detailed description of the Adam algorithm, the reader is referred to [KB14]. Through the introduction of the mean (2.19) and uncentered variance (2.20) of the gradient, Adam deviates from a purely first-order optimization technique and shares similarities with more complex optimization methods. Some of these methods, such as Quasi-Newton Methods, we will discuss in Chapter 4 in the context of image registration.

Since its inception, Adam has established itself as the default optimizer for many deep learning tasks. In recent years the community proposed adjustments to the original formulation to further improve generalization and convergence rates. In 2016 Timothy Dozat argued for the incorporation of Nesterov Momentum based on Nesterov's accelerated gradient, a method well-known in the optimization community [Doz16]. The central idea is to apply the momentum step in (2.19) *before* the gradient update. The author provides only a single example of image compression using a small encoder-decoder network applied to the MNIST data set to validate the faster convergence rate of **Nesterov-accelerated Adam** (NAdam). Therefore this study serves as proof that Adam's performance can be increased further.

The $L_2$ regularization and weight decay are often used synonymously in the deep learning community. The authors in [LH19] point out that this is not true for optimization with adaptive learning rates. As a result, Adam displays poor generalization when used with $L_2$ regularization in comparison to SGD. As a solution, the authors argue to decouple the weight decay from the optimization steps. To achieve decoupling, the weight decay has to be added to the parameter update in (2.21)

$$\theta_t = \theta_{t-1} - \epsilon \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \zeta) + \lambda \theta_{t-1}, \tag{2.22}$$

where $\lambda$ denotes the decay factor. In their work, the authors added a global scaling factor to enable user-defined scheduling of $\epsilon$ and $\lambda$.

One significant drawback of the Adam algorithm is the large variance of gradients in the early stages of training. Liu et al. [Liu+20] just recently identified this problem. Adam is often used in combination with a warmup heuristic or using smaller learning rates at the start, due to the observation that adaptive learning rate methods tend to converge in local optima during early training stages. Both strategies mitigate the phenomena but do not address the underlying problem. Liu and his colleagues attribute the behavior of Adam in the early stages to a large variance, due to the limited amount of samples. To address this they promote **Rectified Adam** (RAdam), an Adam variant limiting the variance. The proposed variation is not only applicable to Adam but to all adaptive learning rate optimizers, such as NAdam [Doz16] or RMSprop [Hin12].

The Adam variants each address a disadvantage of the original process. However, they require a more precise tuning to the task and introduce additional parameters. As we do not intend to analyze different optimizers in this thesis, we will refrain from an application and use vanilla Adam instead.

**Scheduling.** Both the authors of [LH19] and [Liu+20] emphasize the benefits of scheduling, which is the process of global learning rate adaptation during training. In classical optimization line-search algorithms are used to determine the optimal step length, effectively using a different learning rate/step size for each update. One popular representative is the Armijo line search algorithm, which will be discussed in detail in Chapter 4. The graph nature of gradient calculation in most deep learning frameworks prohibits the determination of an optimal learning rate. Instead varying heuristics to ease convergence can be used. They can be rather static, changing the learning rate by a constant or a linearly changing multiplicative factor. Different trigger options are available, e.g. reaching a pre-defined milestone or no improvement of the evaluation metric in a given time window.

Apart from strategies monotonically decreasing the learning rate more elaborated methods surged in recent years. In close connection to adaptive learning rate optimizers like Adam, cyclical rate schedules are proposed in [Smi17]. The idea is to let the learning rate vary between an upper and lower bound to boost convergence. Suitable bounds can be found through the evaluation of a given model with increasing learning rates after a fixed number of epochs. Another interesting technique is so-called warm restarts, described among others in [LH17]. The learning rate is periodically initialized to a

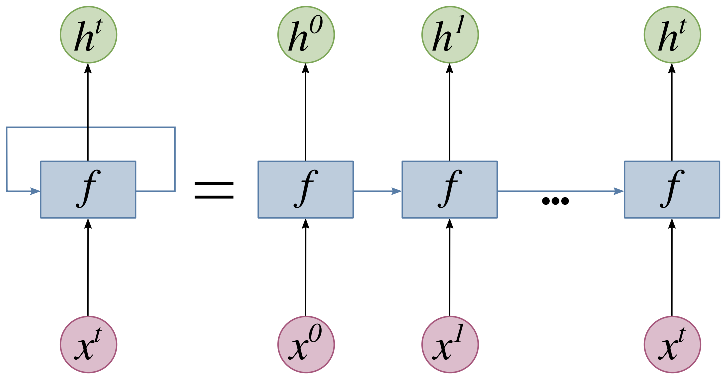


Figure 2.3.: Sketch of a RNN with $t$ sequence entries in unrolled form. The output $o^t$ is utilized as an additional input $h^t$ in combination with the next sequence input $x^{t+1}$.

pre-defined value and decreases in a set time frame. Warm restarts require a quarter of the epochs compared to currently-used learning rate schedule schemes. The decay function can also be adapted, in the original work a cosine annealing is used.

Scheduling established itself as an important tool to speed up convergence and a set of schemes is tested in Chapter 5.

## 2.2. Sequence Modeling

Although the fundamentals of multi-layer neural networks have been known for a long time [LeC+98], their popularity increased due to more complex architectures and advancements in computational hardware. One class of these more advanced model architectures are sequence models, their most prominent members being **Recurrent Neural Networks** (RNN). In contrast to CNNs, which are specialized networks for values on a grid and images, RNNs are designed to effectively process sequential inputs in the form of $\mathbf{x}^{(1)}, ..., \mathbf{x}^{(T)}$. Much like CNNs, RNNs use weight sharing over multiple time steps instead of channels. The concept of RNNs is to introduce and iteratively update an internal state $h^{(t)}$ according to

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta). \tag{2.23}$$

Ideally, the state $h^{(t)}$ captures a summary of the past sequence inputs, enabling the network to keep track of past information to predict some future output.

A visualization of the network described in (2.23) can be found in Figure 2.3. RNN architectures are widely used in the fields of language processing, such as translation tasks, and reinforcement learning. In both areas, the past sequence positions, either words or taken actions, are of great importance for following sequence entries. There are many subcategories of RNN architectures, mainly differ in the connections between the states $h^{(t)}$ and the input sequence positions $x^{(t)}$ or output sequence positions $o^{(t)}$. For a more in depth-view of RNN designs and back-propagation through these networks, the interested reader is referred to [GBC16].

### 2.2.1. Long-range Dependencies

The unrolled RNN in Figure 2.3 hints towards a typical problem when working with these architectures: The recursive structure of RNN models causes them to be very deep, which leads to numerical stability problems in the gradient calculation. In the case of vanishing gradients, it becomes difficult to correctly determine a descent direction for the weight update. On the other hand, learning is also highly unstable with exploding gradients [GBC16]. Accumulation of large gradients results in very large updates to model weights and hinders convergence.

In addition to these difficulties associated with deep network architectures the weights in later layers tend to become small. Bengio et al. [BFS93] demonstrated that the magnitude of the gradient at step $t$ with respect to state $t^0$ decreases exponentially as $t$ increases. In conclusion, the magnitude of long-term interactions in a sequence is exponentially smaller than the gradient of short-term interaction, thus learning the **long-term dependencies** will take more time and often be hindered by minor changes in the short-term interactions. One way to counter the exponential decay of gradient magnitude is adding skip connections, see residual connections in Section 2.1.2. Instead of connecting every step $t$ to its successor state, it is connected to the state $t + \tau$ with a delay tune-able $\tau$. This strategy slightly mitigates the decay problem and allows the network to capture longer dependencies [GBC16]. Another possible solution is the use of gated RNNs, which will be discussed in the next section.

### 2.2.2. LSTM

Gated RNNs are designed to model the long-range dependencies other architecture types struggle with. The core concept is to create a path through the model along which the derivative magnitudes are stable. This is achieved by weighting the connections inside the network, in effect creating a trainable gate at each step $t$. These gates enable the network to autonomously discard information, clearing the corresponding state and thus changing the number of included steps dynamically.

The initial idea to implement an additional direct path through the RNN architecture is attributed to Hochreiter and Schmidhuber in their work **Long Short-Term Memory** (LSTM) [HS97]. The LSTM layer is a small neural network in itself but is used in the same way as any other recurrent layer $f$ in (2.23). There are three gates and one additional state defined in an LSTM layer. All important states, layers, and the flow of information through an LSTM cell are depicted in Figure 2.4. The most important steps are the forget gate (1), input gate (2), current state (3), and output gate (4).

The *current state* $C^t$ is often used as output at the end of a network consisting of $t$ LSTM cells. The current state can be calculated as

$$C^t = v^t \odot C^{t-1} + i^t \odot \tanh(\Theta_c \cdot \left[h^{t-1}, x^t\right] + b_c). \tag{2.24}$$

Here $\odot$ denotes the point-wise multiplication.

It becomes clear that to calculate the current state, the outputs of the other gates are

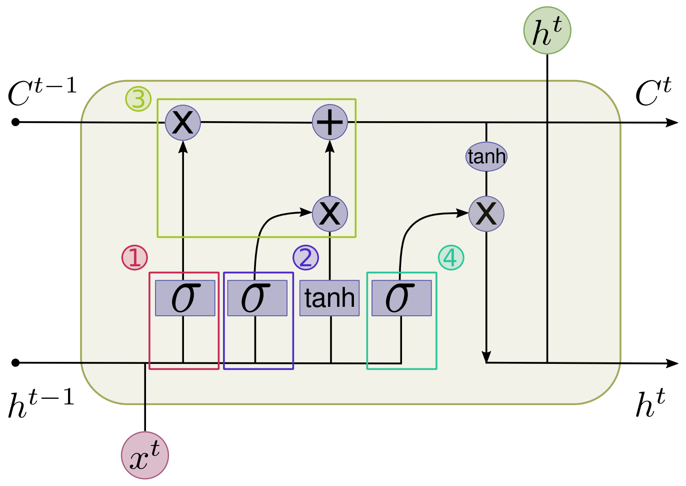


Figure 2.4.: Sketch of an LSTM cell. The grey-blue boxes denote the network layer, where $\sigma$ and tanh denote the activation functions. The circles are point-wise operations (addition + or multiplication x). Where paths meet, a concatenation is performed; at departing positions, the object is copied. The gates are marked as (**1**) forget gate, (**2**) input gate, (**3**) current state, and (**4**) output gate. The current state is only used internally to control the influence of long dependencies in the input sequence $x^t$.

needed. Starting with the *forget gate*

$$v^t = \sigma(\Theta_v \cdot \left[h^{t-1}, x^t\right] + b_v), \tag{2.25}$$

defined as $v^t$ in this thesis instead of $f_t$ in most literature in order to avoid confusion with the layer function $f$. The forget gate controls which information from previous positions is important at the current position. This previous information is encoded in a state variable $C^{t-1}$, the most direct way to pass information within the LSTM network.

The next gate is the *input gate*

$$i^t = \sigma(\Theta_i \cdot \left[h^{t-1}, x^t\right] + b_i) \tag{2.26}$$

analogous to (2.25), but with its own weights and bias. Here the information flowing from the input to the current state is selected.

The *hidden state* $h^t$ enables information from step $t$ to flow through multiple subsequent cells. It is calculated by the *output gate* $o^t$

$$h^t = o^t \odot \tanh(C^t). \tag{2.27}$$

The *output gate* is defined in the same way as forget and input gates with their weights and biases.

The LSTM architecture has gained success in various tasks over the past years. We present some selected applications, which expand on the principle concepts of layer design. Although the LSTM architecture is designed with sequence data in mind, the performance in speech recognition was inferior to deep fully connected networks for a time. One reason is that LSTM architecture is only able to make use of previous entries in the input sequence. In speech recognition, the subsequent sequence entries should also be exploited to achieve the best performance.

**Bidirectional LSTM.** In [GMH13] the authors extend on the original LSTM design in (2.25) to (2.27) and introduce a *bidirectional LSTM*. Two separate hidden layers process the data in both directions, which are then fed to the same output layer. For simplicity, we summarize all LSTM gate functions into one hidden layer function $\mathcal{H}$. The network computes a forward $\overrightarrow{h}$ and backward $\overleftarrow{h}$ hidden sequence

$$\begin{aligned}\overrightarrow{h}^t &= \mathcal{H}(\Theta_{x\overrightarrow{h}}x^t + \Theta_{\overrightarrow{h}\overrightarrow{h}}h^{t-1} + b_{\overrightarrow{h}})\\ \overleftarrow{h}^t &= \mathcal{H}(\Theta_{x\overleftarrow{h}}x^t + \Theta_{\overleftarrow{h}\overleftarrow{h}}h^{t+1} + b_{\overleftarrow{h}})\end{aligned} \tag{2.28}$$

and combines both into one output using trainable forward $\Theta_{\overrightarrow{h}y}$ and backward $\Theta_{\overleftarrow{h}y}$ weight matrix

$$y^t = \Theta_{\overrightarrow{h}y}\overrightarrow{h}^t + \Theta_{\overleftarrow{h}y}\overleftarrow{h}^t + b_y. \tag{2.29}$$

Both formulas in (2.28) are LSTM layers in one sequence direction either front to back or back to front, recognizable by the directional arrow. Both layers have trainable weight matrices $\Theta_{..}$, biases $b_{..}$, and hidden states $h_{..}$. The backward layer is iterated from $t = T$ to 1, the forward layer from $t = 1$ to $T$. This bidirectional extension on the LSTM architecture achieves state-of-the-art results in phoneme recognition [GMH13].

Sequence tagging is another interesting task in natural language processing (NLP). Tagged sequences are used in search engines to recommend web pages or product advertisements. They are the basis for many modern search algorithms. Linear statistical models such as Hidden Markov Models (HMM), Maximum Entropy Markov Models (MEMMs) [MFP00], and Conditional Random Fields (CRF) [LMP01] used to be the best-performing methods for sequence tagging. In [HXY15], Huang et al. apply various LSTM architectures to sequence tagging, including a bidirectional LSTM (2.28, 2.29). Furthermore, they add a CRF layer to the model in [GMH13] to incorporate the established statistical methods. The authors report a consistent improvement in tagging accuracy compared to a single CRF model with identical features.

**Long-Term Recurrent Convolutional Network.** A drawback of the LSTM architecture is its use of fully connected layers, which limits their usefulness for image sequence data. A solution is proposed in [Don+15], where the authors combine a CNN-based feature extraction network with stacked LSTM layers for visual recognition and description in video sequences (see Figure 2.5). Their method is characterized by a deep hierarchical visual feature extractor (CNN) and the temporal depth of recurrent networks (LSTM), rendering the model especially suited to deal with tasks such as video recognition, image description, or video narration. The *Long-Term Recurrent Convolutional Network* (LRCN) can learn to recognize and synthesize temporal dynamics in sequential data.

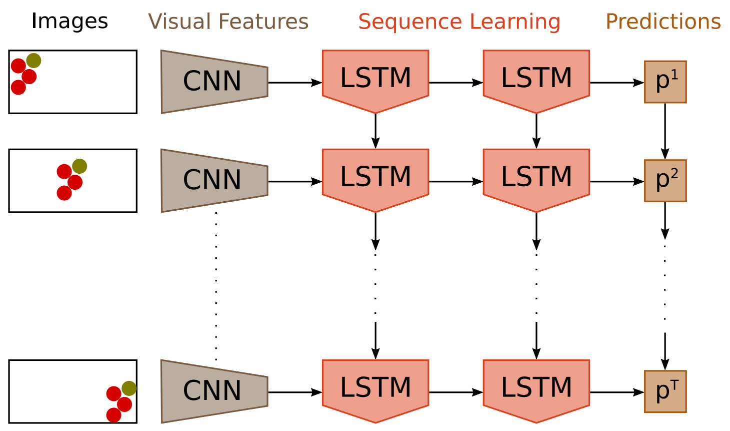


Figure 2.5.: The Long-Term Recurrent Convolutional Network (LRCN) architecture combines the strengths of CNNs' visual recognition and RNNs' ability to model time-varying inputs and outputs. LRCN processes the visual input sequence (**left**) with a convolutional feature extractor (**center left**). These features are passed to a stack of recurrent sequence models (LSTMs, **center right**), which produce a variable-length prediction (**right**) [Don+15].

The LSTM-type architecture improves purely CNN-based models on conventional video activity challenges and enables direct mapping from image pixels to language descriptions. The authors demonstrate that learned sequential dynamics can improve on learning in the visual domain only, as well as learning the dynamics of an output sequence using fixed visual features [Don+15].

**Convolutional LSTM.** Although combining convolutions and sequence learning has its benefits, the principal structure of LSTM layers is untouched. Thus direct application to higher-dimensional data is still unfeasible. In precipitation nowcasting, the goal is to predict future rainfall intensity based on past radar maps. Given a sequence of several past radar maps as inputs, the desired output is a sequence of future radar maps. The forecasting problem contains both spatial and temporal information relevant to precise prediction. A solution to this problem is the *Convolutional LSTM Network* (ConvLSTM) [Shi+15]. The authors replace all fully connected matrix multiplications in (2.25) to (2.27) with convolutional operations

$$\begin{aligned} v^t &= \sigma(\Theta_{xv} * x^t + \Theta_{hv} * h^{t-1} + b_f) \\ i^t &= \sigma(\Theta_{xi} * x^t + \Theta_{hi} * h^{t-1} + b_i) \\ C^t &= v^t \odot C^{t-1} + i^t \odot \tanh(\Theta_{xc} * x^t + \Theta_{hc} * h^{t-1} + b_c) \\ o^t &= \sigma(\Theta_{xo} * x^t + \Theta_{ho} * h^{t-1} + b_o), \end{aligned} \tag{2.30}$$

where $\Theta_{..}$ now denotes kernel weights, and where $*$ is the convolution operator. The new ConvLSTM is able to handle spatio-temporal data by encoding spatial information, which the full connections in input-to-state and state-to-state transitions in the original LSTM layer are not. In addition, other architectural designs such as Encoder-Decoder structures can also be utilized. The authors report an improved performance on synthetic and real-world data in comparison to a state-of-the-art optical flow based algorithm, proving that the advantages of sequence models can be preserved and also be applied to image data [Shi+15].

As demonstrated above, the LSTM layer is an interesting and versatile model with many applications in sequence modeling and prediction. It is flexible enough to enable adaptation to other tasks, where the data itself is not primarily a one-dimensional sequence but expresses some sequence-like attributes. This suggests that it can be useful in an iterative process commonly occurring in an optimization context, which we will investigate in Chapter 5.

## 2.3. Transformer Networks

The problem of capturing long-term dependencies in RNNs (Section 2.2.1) is not yet satisfactorily solved and very long sequences remain a challenge. In recent years, a new architecture, the Transformer [Vas+17], gained popularity, due to its success in processing long sequences and modeling long-term dependencies. The idea of building loops into a network is discarded for the mechanism of attention, focusing on smaller partitions of the input while processing large amounts of information. Attention in machine learning and especially neural networks can take various forms, from visual attention [Xu+15] and encoding strategies [SVL14], to weighting the relation between positions in a sequence [Vas+17]. We briefly summarize the latter, as it is most relevant to the remainder of this thesis.

### 2.3.1. Attention

Attention serves as a method to allow the network to focus on certain positions in a sequence. It enables the model to weigh the relationship between some or all entries within a sequence. In language processing attention enables the model to learn if certain words occur regularly in conjunction and even to keep track of connections over multiple sentences.

Attention-based models have sparked increased interest in recent years. They boost performances of neural networks in a variety of tasks such as handwriting synthesis [Gra13], machine translation [BCB15], image caption generation [Xu+15], and object classification [MHG+14]. The attention mechanism is formulated as a function that produces a weighting factor $z^i$ for a given sequence $s$ of length $L$, scoring the sequence entry $s^i$

$$\begin{aligned} z^i &= \mathrm{Score}(s^i) \\ \mathrm{Attention}(z^i) &= \frac{e^{z^i}}{\sum_{j=1}^{L} e^{z^j}}. \end{aligned} \tag{2.31}$$

The *Score* function in (2.31) can be designed in a multitude of ways. Thus attention is a highly flexible concept, where the exact formulation of *Score* varies.

Attention($z^i$) can be interpreted as a hidden state decoding the relevant relations between the input position $z^i$ and all other entries in the sequence. This method is easy to use in combination with RNNs. Each input position has an attached hidden state decoding the importance of each position in the sequence for the resulting translation. The hidden state is composed of matrix rows displaying the influence of the corresponding input position to each entry in the output sequence. The hidden states can be attained in different ways, for example, learned during training. It can be seen, that for longer sequences these matrices become quite large and computationally expensive. Additionally, processing of inputs in parallel is impossible, rendering it unfeasible for our long input sequences.

**Visual Attention.** Much like the ConvLSTM in Section 2.2.2, extending attention beyond one-dimensional sequences is of particular interest. Most notable is the work of caption generation in [Xu+15]. Instead of sequence data, the authors utilize a convolutional extractor network to create $d$-dimensional representations for $L$ image parts. An LSTM layer creates the caption sequence by generating a word-based context vector, the previous hidden state, and the previously generated sequence entries. For the score function, a multilayer perceptron $f_{\text{att}}$ conditioned on the previous hidden state is used. Two different mechanisms for $f_{\text{att}}$ are discussed: a *soft* deterministic and a *hard* stochastic variant, for more details see [Xu+15]. The proposed attention-based approach provides state-of-the-art performance on the three benchmark data sets Flickr, METEOR, and MS COCO. The authors also demonstrate that the learned attention can be exploited to give more interpretability by visualizing the attention weights in the original image.

**Location Attention.** Speech recognition is a more difficult task in the field of NLP. It requires longer input sequences — thousands of frames instead of a few words — as well as the ability to differentiate multiple similar speech patterns. Although there exist suitable models, they often require dictionaries of hand-crafted pronunciation and phoneme lexica to deal with long speech patterns. In [Cho+15], an attention-based network is proposed. The authors tackle the problem by adding location attention to the content-based attention mechanism in [BCB15]. This is achieved by adding features to the attention mechanism derived from convolving the attention weights from the previous step with trainable filters. By doing this, the authors decrease the baseline model's [BCB15] error rate by 1%, elevating it to the performance level of dominant speech recognition models.

Attention has become a vital tool in deep learning. It improves prediction accuracy on difficult real-time detection tasks [Sch+18]. Apart from convolutional networks, it is also possible to incorporate attention into other model types, such as graph networks [Vel+17; Fuc+20]. Another interesting work focuses on leveraging the attention mechanism beyond categorical distributions using graphical models [Kim+17]. A more thorough analysis of attention in deep learning is given in [NZY21].

**Scaled Dot-Product Attention.** In this thesis, we employ scaled dot-product attention. The Transformer architecture provides a more favorable attention mechanism than the hidden state formulation in Section 2.3.1. Following the definition in [Vas+17], the attention function can be described as mapping a query vector and a set of key-value vector pairs to an output. The authors named their attention function *Scaled Dot-Product Attention*, computing the dot or inner product of the query and key vectors. Query and key vectors have the dimension $d_k$, whereas the values are of dimension $d_v$. To generate these vectors, the embedded sequence $x$ is multiplied by the three trainable matrices $W^Q$, $W^K$, and $W^V$. For each position $x_i$ in the sequence $x$ we get a query $Q_i = x_i W^Q$, key $K_i = x_i W^K$, and value $V_i = x_i W^V$. Multiple queries, keys, and values are packed in the matrices $Q$, $K$, and $V$ to apply the attention function simultaneously. The output is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \ , \tag{2.32}$$

where the softmax is a special case of an activation function $\sigma_s : \mathbb{R}^K \to [0, 1]^K$:

$$\sigma_s(\mathbf{z})^i = \frac{e^{z^i}}{\sum_{j=1}^{K} e^{z^j}} \tag{2.33}$$

with $i = 1, .., K$ and $\mathbf{z} = (z^1, ..., z^K) \in \mathbb{R}^K$. Compared to our definition of the attention mechanism in (2.31), the normalization $\sigma_s$ is exactly the same, but the score function is changed to an inner product of trainable vectors. To further increase the effectiveness of attention, the authors suggest [Vas+17] to perform linear projections of queries, keys, and values to the respective $d_k$ and $d_v$ dimensions.

These projections can be implemented using a fully connected layer in form of matrix multiplication. These projections are trainable and the attention function is applied to each projected version of queries, keys, and values in parallel. They call this multi-head attention, defined by

$$\begin{aligned} head_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \\ \text{MultiHead}(Q, K, V) &= \text{Concat}(head_1, ..., head_h)W^O. \end{aligned} \tag{2.34}$$

Where $W_i^Q$, $W_i^K$, $W_i^V$ are trainable matrices describing the projections of queries, keys, and values and $W^O$ is a trainable matrix to project $h$ heads to a combined output.

## 2.3.2. Transformer Architecture

The principle components of the Transformer can be seen in Figure 2.6, introduced in [Vas+17]. The most noticeable characteristic is the twofold structure consisting of an encoding part and a decoding part. This Encoder-Decoder structure is widely applied in the field of computer vision. Renowned examples are Variational Autoencoders (VAE) [KW13] and Generative Adversarial Networks (GAN) [Goo+20].

Since Transformers are designed to process language, the first layer is an embedding. Natural language cannot be processed directly by neural networks. Therefore, texts

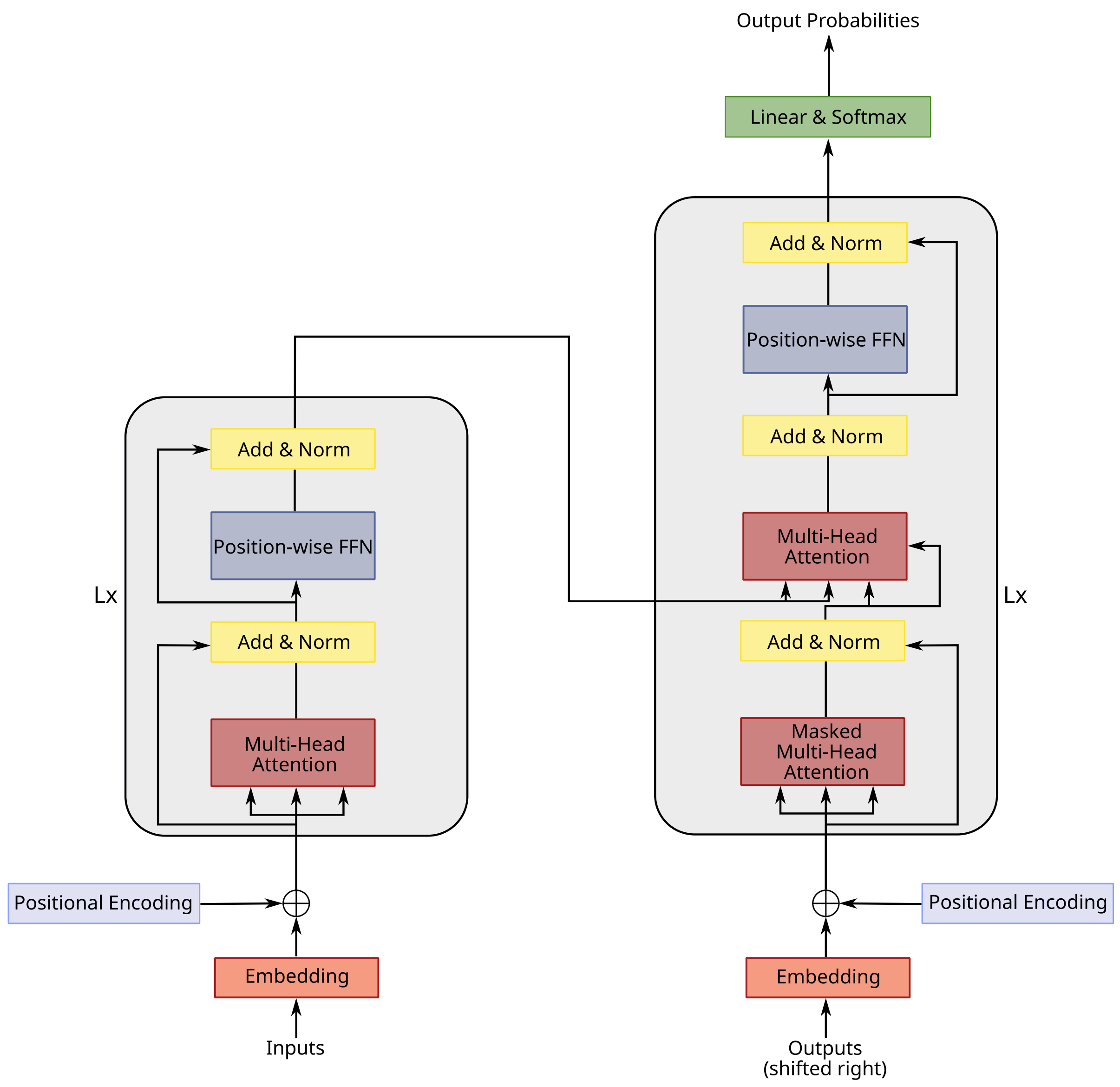


Figure 2.6.: The Transformer network consists of an encoding part (**left**) and a decoding path (**right**). Both contain attention, fully connected, and normalizing sub-layers. Within each part, residual connections are applied between the attention and fully connected sub-layer. The encoder and decoder can consist of repeated sub-layer building blocks, indicated by the gray background panel [Vas+17].

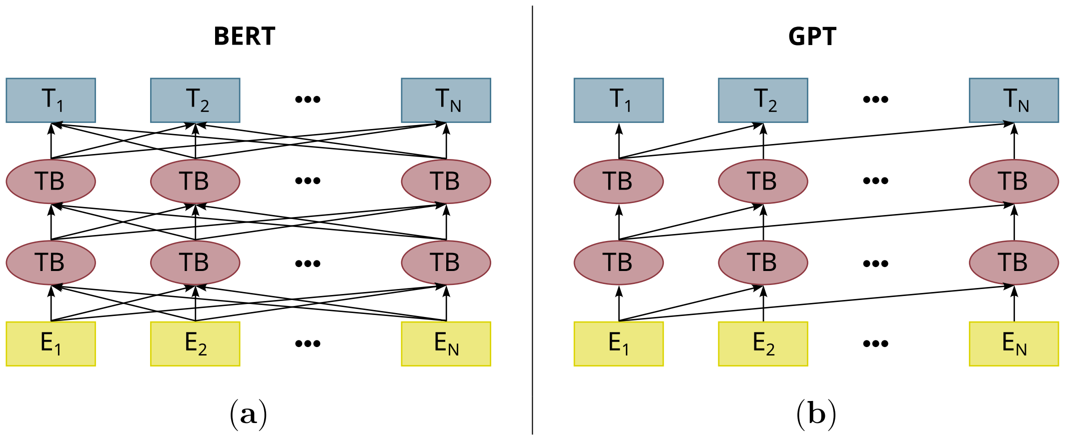


Figure 2.7.: Two pre-training model architectures with a fine-tuning approach. The **output sequence** **T**={$\mathrm{T}_1, ..., \mathrm{T}_\mathrm{N}$} is generated from the embedded **input sequence** **E**={$\mathrm{E}_1, ..., \mathrm{E}_\mathrm{N}$} by means of **Transformer blocks** (TB), see Figure 2.6 for details. (**a**) BERT uses a bidirectional Transformer without a mask in the self-attention mechanism. (**b**) OpenAI's GPT model uses a left-to-right Transformer more closely related to the original model. As a result, BERT models are able to jointly learn both left and right context, which is beneficial for language tasks, such as question answering [Dev+18].

are converted into so-called tokens. Tokens can be individual words, partial words, or characters. The words are often converted into a machine-readable form, for example, binarized, using a fixed vocabulary. Embedding is placing the tokens into the corresponding vocabulary, a vector of variable size $d_{\mathrm{model}}$. Instead of a fixed vocabulary, here embedding is trainable using a linear layer.

Since the Transformer model does not contain convolutions and recurrences the information about the position of the tokens within the sequence must also be coded. The *positional encoding* adds a value after embedding, which depends on the token's position in the sequence. Many variants of *positional encoding* exist, either fixed or trainable [Geh+17]. The main building block for the encoder can be seen on the right side in Figure 2.6 consisting of a multi-head attention and a feed-forward part, a fully connected layer. There are residual connections for both the attention and feed-forward part, adding the unprocessed input to the output.

This main block can be repeated multiple times to form the encoder. The decoding path is created similarly but contains an additional masked multi-head attention. This is necessary to block information of subsequent positions leaking into prediction and ensures that the model at position $i$ can only attend to outputs of positions less than $i$. The decoder block can also be repeated multiple times. It is possible to deploy encoder- or decoder-only variants of Transformers if the output does not have to be a sequence.

**BERT.** Transformer models [Vas+17] have demonstrated dominant performance on a broad range of sequence tasks, including machine translation, question answering, or text classification. Two popular models are the *Bidirectional Encoder Representations from Transformers* (BERT) architecture [Dev+18], which is able to attend to previous as well as subsequent entries in a sequence, and the unidirectional GPT [Ram+21] (Figure 2.7). Both models already can be considered large with 110 and 340 million parameters, but the trend is towards even larger designs such as the GPT-3 [Bro+20] model with 175 billion parameters. Models of this size rely heavily on hyperparameter choice [Liu+19], but extensive parameter search is especially time and memory consuming.

**Sparse Transformer.** As memory is a constraint for most research groups and the application of ever larger models is not feasible in many everyday tasks, much work is done to reduce the size and memory consumption. This led to the development of *Sparse Transformer* models [Chi+19]. The contribution of this work is three-fold: dealing with weight initialization, sparse attention kernels, and recalculation of weights during the backward pass. We will discuss the sparse attention kernels in greater detail and refer the interested reader to the original work for more information on the other topics. One reason for the high memory demand of Transformer models is the dense attention matrix. In the full self-attention mechanism, $n \times n$ weights are computed for a sequence of length $n$. Memory consumption grows quadratically with the sequence length. In [Chi+19] the authors propose to separate the full attention computation into several faster operations, which combined yield the full dense operation.

Following the definition in [Chi+19], we extend (2.32) and parameterize the matrices by a connectivity pattern $S = \{S_1, ..., S_n\}$, where $S_i$ denotes the indices of the input sequence to which the $i$-th output attends:

$$\begin{aligned} \text{Attention}(X, S) &= (a(x_i, S_i))_{i \in \{1,\ldots,n\}} \\ a(x_i, S_i) &= \text{softmax}\left(\frac{(W^Q x_i)(W^K x_i)_{S_i}^\top}{\sqrt{d_k}}\right)(W^V x_i)_{S_i} \end{aligned} \tag{2.35}$$

The output at position $i$ is given by the sum of the values weighted by the dot-product of the keys and queries. $X$ is a matrix of embedded inputs and $W^Q$, $W^K$, and $W^V$ are again the weight matrices to create the queries, keys, and values.

For standard dense left-to-right attention, every element can attend to itself and all previous elements with the connectivity pattern $S_i = \{j : j \leq i\}$. The proposed factorized self-attention utilizes $p$ separated attention heads, each a subset of indices. The $m$-th head $A_i^{(m)} \subset \{j : j \leq i\}$ is again the set of indices the $i$-th output attends to. Two criteria are used to choose suitable subsets. All sets should be relatively small $|A_1^{(m)}| \propto \sqrt[p]{n}$ and enable all input positions to attend to all output positions in $p$ steps. This reduces the computational cost to $O(n\sqrt[p]{n})$. Two different factorization schemes for $p = 2$ are tested: Strided attention for periodic data and fixed attention for sequences with arbitrary correspondences.

In [Chi+19] the sparse Transformer model is tested on autoregressive sequence generation. Images, text, and audio are treated as sequences using discrete tokenization, i.e., raw byte representation. Using these sparse attention patterns, the augmented models achieve state-of-the-art performance for compression and generation of natural

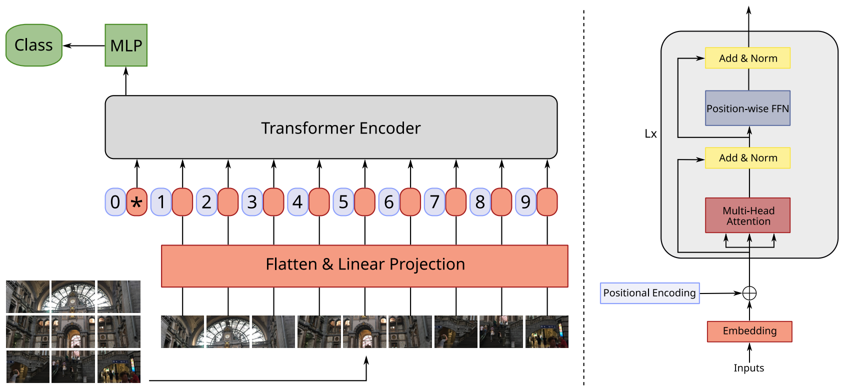


Figure 2.8.: **Left**: Overview of the ViT model [Dos+21]. An image is split into fixed-size patches. These patches are **linearly embedded** and the **position** in the image is encoded. The resulting sequence of vectors is passed into a standard Transformer encoder. An additional trainable classification token * enables a small MLP model to classify the patch sequence in a supervised setting. **Right**: Standard Transformer encoder block, here we kept the embedding and positional encoding of the original architecture. Note that these steps are substituted by the corresponding projection layer and positional encoding in the ViT model.

language, audio, and images. Although Transformer networks achieved great success in dealing with sequence data, the application to computer vision tasks was limited at the beginning. Frequently, attention mechanisms and CNNs are used in conjunction. But lately, transformers are applied directly to image data.

**Vision Transformer.** An interesting work is that of Dosovitskiy et al. called *Vision Transformer* (ViT) [Dos+21]. A visualization of the model can be found in Figure 2.8 on the left side. Similar to LRCN architecture in Section 2.2.2, they aim to adapt the sequence model to image data. In contrast to LRCN, Dosovitskiy and his colleagues do not combine convolutional and Transformer networks, but use a pure Transformer model, preprocessing the image data before the application of the model: An image is split into small patches followed by a linear embedding of these patches. The input for a standard Transformer encoder-only model is a sequence of these embedded image patches.

The computational cost of the attention matrices grows quadratically with the sequence size, as described in the context of sparse Transformers. Simple flattening the image and applying attention pixel-wise is therefore not feasible. Instead, the authors reshape the images $x \in \mathbb{R}^{H \times W \times C}$ into a flattened sequence of 2D patches $x_p \in \mathbb{R}^{N \times (P^2 \dot{C})}$, where $(H, W)$ is the image resolution, $C$ the number of channels, and $(P, P)$ the resolution of the resulting patch. The number of patches corresponds to the input sequence length for the Transformer model. The latent space is set to fixed size $D$ and all flattened patches are mapped to the latent space using a trainable projection layer. The problem of image size and computational cost could also be tackled using a variant of sparse Transformers, however, these require complex engineering to be implemented efficiently.

To enable classification using the Transformer encoder model, a trainable token is added to the start of the sequence, marked in Figure 2.8 by the asterisk. This token serves as an image representation at the output of the Transformer encoder. A classification head parameterized by a few-layer MLP determines the class distribution based on the representation token (see Figure 2.8). A multitude of position encodings exist, e.g., fixed periodic encoding using a sine/cosine function [Vas+17] and learned encoding [Geh+17]. In the case of the ViT model, the authors used trainable position encoding.

Dosovitskiy et al. also study the performance/compute trade-off for both ViT and ResNet. In their experiments, they discover that ViT attains the same level of performance as ResNet with approximately $2 - 4x$ less computing power. The hybrid model slightly outperforms ViT adding a small faction to the computational costs due to the feature extraction network. In summary, Vision Transformer displays state-of-the-art performance on many image classification tasks, while being relatively cheap to pre-train.

Due to their many desirable properties and impressive performances, Transformers are used in a wide spectrum of applications with many different model designs. The survey in [Kha+21] provides a great overview of recent developments.

All of these models have beneficial properties for the analysis of MALDI-MSI data (Chapter 3). Bidirectional Transformers, such as BERT, link positions in both directions of a sequence. MALDI-MSI spectra are not directional sequences. Although the position within the spectrum provides information about the mass of the measured molecular structures, correlations are possible between high and low mass structures as well as between structures of similar mass. This property can be accounted for using bi-directional transformers. Instead of using classical dimensionality reduction methods to enable the usage of models such as the Transformer, *Sparse Transformers* could be applied directly to the spectra. In addition to the sparsity patterns presented here, a pattern could be developed that takes into account the structure of MALDI-MSI spectra. Similarly, a 1D-convolution variant of ViT could be used to generate much shorter sequences from the spectra. These would consist of only a few tokens.

However, we first start with an encoder-only Transformer using the initial design in [Vas+17] to evaluate the utility of the Transformer model in the context of MALDI-MSI (Chapter 3).

## 2.4. Meta-Learning

The above network architectures are designed to be trained for solving a single task, e.g., mapping objects to a finite set of classes. Meta-learning forgets the learning process itself, intending to derive learning algorithms from scratch. Meta-learning networks can be used to find a favorable initialization of another neural network or update the network parameters based on the loss function.

This results in a typical "bi-level" structure: The "supervisory" meta-learner network (*optimizer*) modifies the underlying solver algorithm (*optimizee*) and guides it to fitting solutions of the corresponding task-specific network. In this thesis, we adopt some popular

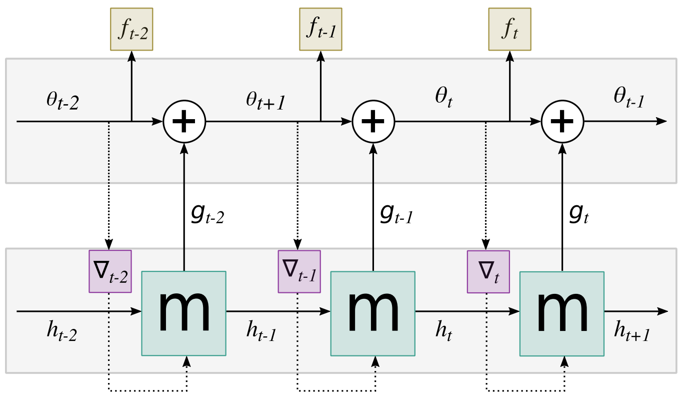


Figure 2.9.: Computational graph for (2.37) used in [And+16] to compute the gradient of the *optimizer*. **Top**: Flow in the *optimizee* to increase its performance. **Bottom**: Flow in the LSTM cells (*optimizer*), depicted are the hidden states $h_t$ and the input $\nabla_t$. This scheme can also be used to introduce nonlinear updates to iterative methods in the field of image registration (Chapter 5). The *optimizer* is also a model based on LSTM layers. The iterative optimization procedure corresponds to the *optimizee*. As it is not a neural network, the weights $\theta$ are not adjusted, but an update for the procedure is returned directly.

strategies of meta-learning to build an iterative network scheme for image registration.

The concept of meta-learning is well-established in the field of machine learning, mostly in the form of evolutionary models, a group of optimization algorithms inspired by natural evolution mechanisms [Tel+21]. Hochreiter et al. [HYC01] expand on these ideas by introducing neural networks. In their work, they prove that recurrent networks, such as LSTM models, are a feasible choice for meta-learning systems and enable computations for large-scale systems previously prohibited by the restricted parameter sizes for non-network-based models. The authors conduct experiments for semi-linear and quadratic functions with a shallow LSTM architecture and are able to derive an algorithm able to approximate the tested functions.

In [And+16] Andrychowicz et al. propose to replace the manual tuning with learned update rules so that the architecture of the solver is less problem specific and can be easily adapted to new tasks. Given an objective function $f(\theta)$ defined over some domain $\theta \in \Theta$, the classical training approach for differentiable functions is some form of gradient descent:

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t). \tag{2.36}$$

Modifying (2.36) by introducing a learned update $g_t$ with its own set of parameters $\phi$, yields

$$\theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \phi). \tag{2.37}$$

The computational graph of (2.37) is shown in Figure 2.9. In [And+16], the authors use

RNNs to explicitly model the update rule $g_t$. They achieve this by using a coordinate-wise LSTM optimizer consisting of only two layers. They tested their method on different tasks, including CIFAR-10 classification and style transfer. The LSTM optimizer outperformed stochastic gradient descent (SGD) and the popular Adam method, among other methods.

**MAML.** The popular *Model-Agnostic Meta-Learning* (MAML) architecture was introduced in [FAL17]. The authors proposed a model-agnostic algorithm compatible with any model trained with gradient descent. In contrast to the aforementioned models, it does not learn an update function or learning rule. The network uses entire tasks as inputs and adapts to new ones in a few-shot manner. It shows excellent generalization to new tasks with only a few task-specific updates to produce well-performing weight initialization for any underlying gradient-based model. The authors define such a model as a parameterized function $f_\theta$ with parameters $\theta$. For a new task $T_i$, the model's parameters are denoted by $\theta'_i$. A gradient update for task $T_i$ takes the form of

$$\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{T_i}(f_\theta) \tag{2.38}$$

with a fixed or learned step-size $\alpha$. Given a set of tasks $p(T)$, a meta-learning objective can be formulated,

$$\min_\theta \sum_{T_i \in p(T)} \mathcal{L}_{T_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{T_i}(f_\theta)}), \tag{2.39}$$

and the model parameters $\theta$ are optimized in a few update steps. Similar to classical neural network models the meta-learner's weights are updated using SGD across all tasks in $p(T)$. In their work, the authors demonstrate state-of-the-art performance in regression, classification, and reinforcement tasks using this few-shot meta-learning model.

**Meta-Curvature.** Expanding on the MAML model, meta-learning can also be used to elevate the performances of classical gradient-based methods. In [PO19] Park and Oliva define trainable curvature matrices which can be applied on an input gradient tensor. They state that learning curvature information improves generalization even further than the first-order method in [FAL17]. They claim that given multiple tasks there exists some common curvature information and propose a method to precondition the gradients by a learned curvature matrix before updating the underlying model during training. This approach is related to classical second-order methods such as Newton's method, which we will discuss in greater detail in Section 4.3. In order to achieve this, the authors define a meta-curvature function applied to a gradient tensor $\mathcal{G} \in \mathbb{R}^{2\times 3\times d}$

$$\mathrm{MC}(\mathcal{G}) = \mathcal{G} \times_3 M_f \times_2 M_i \times_1 M_o, \tag{2.40}$$

with $M_f \in \mathbb{R}^{d\times d}$, $M_i \in \mathbb{R}^{C_{\text{in}}\times C_{\text{in}}}$, and $M_o \in \mathbb{R}^{C_{\text{out}}\times C_{\text{out}}}$ as trainable parameters. Here $d$ denotes the filter size, $C_{\text{in}}/C_{\text{out}}$ the in and out channel size, and $\times_i$ is the tensor-vector product in $i$-th dimension. Each matrix models dependencies in the input on different levels, namely in one channel ($M_f$), in one filter ($M_i$), and between all filters ($M_o$). The formulation in (2.40) results in a transformed gradient, that is then passed to a given optimizer. The gradient update for the meta-learning objective in (2.38) can be extended to

$$\theta_i' = \theta - \alpha M_{\mathrm{mc}} \nabla_\theta \mathcal{L}_{T_i}(f(\theta)), \tag{2.41}$$

with

$$M_{\mathrm{mc}} = (M_o \otimes I_{C_{\mathrm{in}}} \otimes I_d)(I_{C_{\mathrm{out}}} \otimes M_i \otimes I_d)(I_{C_{\mathrm{out}}} \otimes I_{C_{\mathrm{in}}} \otimes M_f) \tag{2.42}$$

a meta-trained matrix. Here $I_k$ denotes the $k$ dimensional identity matrix. Interestingly the authors state that $M_{mc}$ does not need to be positive-semi-definite to improve generalization. Meta-Curvature leads to faster adaptation and better generalization on synthetic and real-world data and outperforms all tested MAML variants.

For a more comprehensive overview of the developments in meta-learning the reader is referred to [Hos+21]. In this thesis, we focus on updates in the fashion of first-order formulations similar to (2.37) and (2.38).

# Part I.

# Deep Learning for MALDI Mass Spectrometry Imaging

# 3. Deep Learning with Applications to MALDI Mass Spectrometry Imaging

Matrix Assisted Laser Desorption/Ionization mass spectrometry imaging (MALDI-MSI) provides high-resolution molecular information, setting it apart from conventional histological imaging techniques. This makes it a promising technique for histological examinations and an important component of personalized medicine. However, there are still unresolved challenges such as undetected noise sources or the size of the generated spectral data, which makes it difficult to use the full spectral range. These specific challenges of the MALDI-MSI method are described in Section 3.1, alongside a brief introduction including important attributes of the measurements. Recent years brought many advancements in terms of data acquisition and tissue preparation but fewer improvements on the algorithmic classification side. An overview of the current developments of machine learning algorithms in MALDI-MSI is given in Section 3.2. A major part of this thesis is the adaptation of neural networks to the new domain of MALDI-MSI. So far, these methods are little used in the field of MALDI, although they have many useful properties to overcome the given challenges. We analyze the developed methods in Section 3.3 including network architectures, preprocessing steps, and established benchmark methods. Finally, in Sections 3.5, 3.6, and 3.7, we present results for the application to different use cases in the context of MALDI-MSI, followed by a brief discussion of the advantages neural networks bring to MALDI-MSI.

## 3.1. MALDI-Time of Flight

The MALDI-MSI technique allows for the combined analysis of morphological features and protein expression in tissue and enables spatially resolved tissue assessment [Kle+14; Wal+08]. This method enables the determination of spatial distributions of proteins, peptides, and lipids in a single measurement with a spatial resolution of up to 20 $\mu m$. In the following, we provide a brief introduction to data acquisition. In order to better understand the challenges related to the physical processes during data generation, as well as the properties setting mass spectrometry apart from well-known imaging techniques such as computer tomography (CT) or magnetic resonance tomography (MRT). For a detailed description of MALDI-MSI physics, see [Gro13, Chap.11].

The MALDI-MSI pipeline starts with applying a chemical matrix to the sample, a tissue stamp to be examined, and taken from a patient, which is then ionized using a laser to fracture it. The created ions are accelerated in an electric field and picked up by a time-of-flight (TOF) detector. Each fragment has an electrical charge of one,

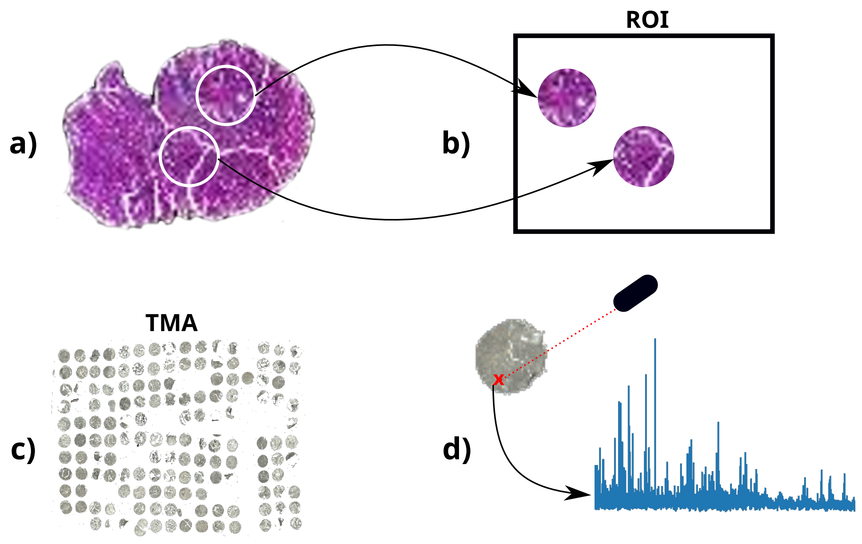
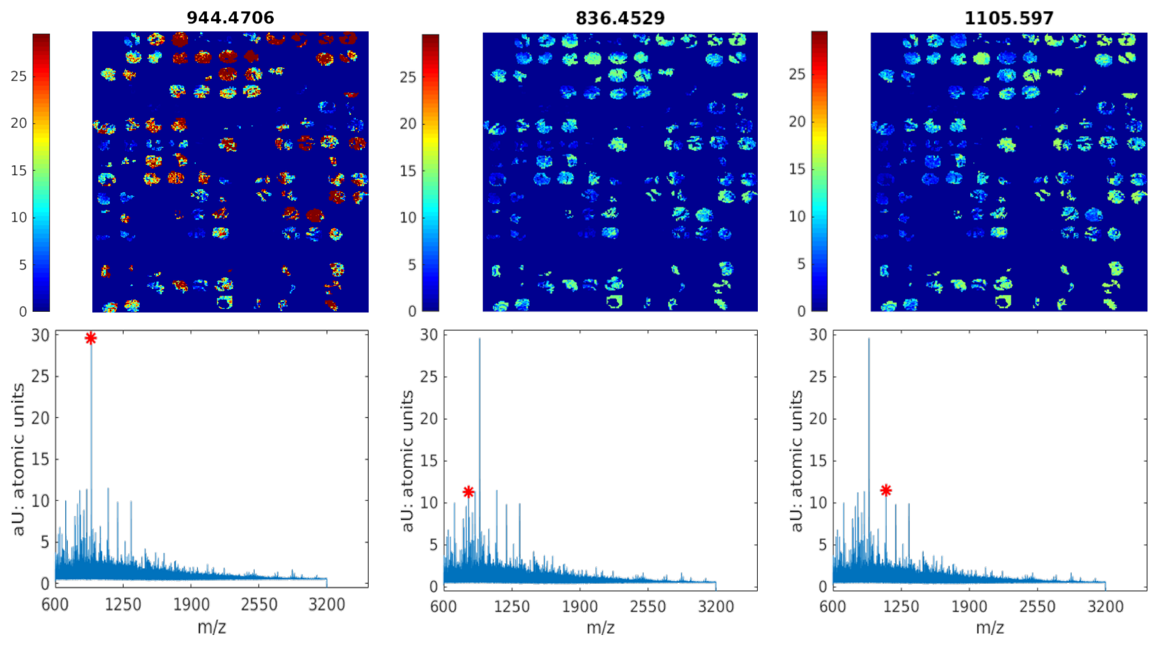


Figure 3.1.: Pipeline of MALDI-MSI data acquisition. (**a**) Biopsy taken from the patient. (**b**) Identification and physical extraction of tumor region of interest (ROI). (**c**) Compiling multiple ROIs from different patients into one Tissue microarray (TMA). The TMA is then prepared for data acquisition. (**d**) MALDI-MSI imaging: A laser fragments the tissue at several points (spots) **x** (red) and a time-of-flight detector measures the molecular mass/charge distribution in the form of a spectrum (blue).

and ions can be separated according to their mass per charge ratio ($m/z$). Ions with lower mass are accelerated more strongly and exhibit shorter detection times. For each sample, a mass spectrum is recorded at a grid of positions, resulting in many spectra per sample. The number of grid points and therefore individual spectra range from $10^5$ to $10^6$ on a given grid. At each location, a spectrum with $10^3$ to $10^4$ $m/z$ values is acquired, depending on the chosen resolution. The magnitude of a single $m/z$ value in a spectrum indicates the relative frequency of the corresponding sample fracture.

To create large data sets multiple samples are combined on a *Tissue microarray* (TMA) [Cas+17]. The technique facilitates the rapid translation of molecular discoveries to clinical applications. The array is constructed by arranging the patient tissue stamps on a carrier plate. In general, multiple samples are taken from different patients with cropped-out regions of interest (ROI) containing various sections of pathological tissue. These are placed onto the carrier plate, which is inserted into a MALDI-MSI instrument, where the samples are processed as described above. Figure 3.1 displays the pipeline of data acquisition used for all experiments in Sections 3.5 to 3.7.

MALDI-MSI can be used to generate spatially resolved molecular signatures (e.g., proteins, peptides, lipids, and molecules of cell metabolites) directly from tissue sections [CFG97; Sto+01; WAD11]. Fixing a single $m/z$ ratio and observing the spatial distribution of its spectral intensity, these can be represented as an intensity image (Figure 3.2). Multiple works utilize so-called $m/z$-images for later processing steps



(a) (b) (c)

Figure 3.2.: **Top**: $m/z$-images for the ovarian cancer data set (Section 3.5). Each image displays all four subtypes. **Bottom**: The corresponding mean spectra for one selected subtype of ovarian cancer. $m/z$-images are heat maps visualizing the magnitude of selected $m/z$ values over the whole data set. Positions in the spectrum can be directly linked to identified proteins in a given tissue. The classification of ovarian cancer subtypes is especially challenging because a limited set of peaks is not specific to any class. The $m/z$-images for 3 $m/z$ values with the largest magnitudes serve as an example. The peaks corresponding to (**a**) 944 $m/z$, (**b**) 836 $m/z$ and (**c**) 1105 $m/z$ are prevalent in each class. Each quarter (vertical) of the image displays samples from one class, see also Figure 3.8.

such as semantic segmentation [LeC+21], clustering [VCV20], or feature extraction with pre-trained neural networks [Zha+20]. This approach drastically reduces data complexity and allows to use tools developed for 2D images. Furthermore, the influence of all noise within a single spectrum on later processing steps vanishes. On the downside, the user has to determine a limited number of peaks to create the $m/z$-images and consequently loses information. Peaks are multiple $m/z$ positions that are characterized by particularly high magnitudes and can be assigned to specific molecular structures such as proteins or lipids. As a rule, such a peak consists of three $m/z$ positions, which describe different isotopes of the same substance. We aim to use the full spectral data as features for subsequent machine learning methods. Few works focus on a similar approach, with [Beh+17; Mit+21] being notable exceptions.

**Noise Sources.** Different *noise sources* are prevalent in MALDI-MSI, which can lead to degraded spectra, posing additional challenges for machine learning methods. MALDI-MSI spectra typically exhibit:

1. an intense and variable chemical noise background, resulting in increased signal intensities [Nor+07].
2. All spectra are recorded independently and magnitudes at the same $m/z$ position may vary for similar metabolic concentrations. Peak magnitudes are estimates of the abundance of similar molecules and can contain errors due to noise artifacts such as ion suppression and electronic noise [Dei+11].
3. Systematic artifacts during acquisition lead to affected mass spectral intensities, resulting from crystal distribution or ion source contamination. Ion transmission may gradually decrease during the time of acquisition, due to chemical inhomogeneities like pH gradients.

Most of these artifacts are related to tissue preparation and vary between acquisitions. For this reason, data sets across different acquisitions present a particular challenge [Dei+11], as noise may produce global ion suppression, meaning that spectra may not reflect actual concentrations and differ vastly from similar spectra with less suppression. In order to take some burden off the machine learning stage, we include a basic preprocessing step that corrects for some of these effects, see Section 3.3.

## 3.2. Related Work

In [Sto+01], Stoeckli et al. argue for more frequent use of MALDI-MSI as a diagnostic tool in clinical practice, in applications ranging from tumor detection to protein identification. The authors highlight the high resolution of the technique as well as the fact that pathological areas (different morphological structures) are often expressed by highly specific signals. Multiple peptide image maps can be created by a single acquisition and enable detailed molecular analysis. Casadonte and Caprioli [CC11] propose a new protocol yielding a speedup in tissue preparation, enhanced ease of use, and more reliable reproducibility.

MALDI-MSI has been introduced in other clinical fields such as histopathology, where it serves as a complementary method to established staining techniques [Lon+16]. Studies commonly focus on specific use cases, such as tumor detection of amyloidosis, while relying on a standardized processing pipeline [Win+17]. For a detailed overview of the current state of MALDI-MSI and recent applications, the reader is referred to [Wal+08].

Besides multiple desirable properties, MALDI-MSI data contains large noise artifacts derived from acquisition and tissue preparation. As this is a major drawback for automatic evaluation, multiple studies were conducted concerning noise reduction.

A strong argument for the necessity of sufficient noise reduction in MALDI-MSI is made in [Dei+11]. The authors tested various normalization techniques such as p-norms or the median filter to reduce the variance in signal intensities. They claim that the popular total ion count (TIC) normalization is a special variant of p-norm and achieves the best results in generating images without noise. But TIC normalization relies on the exclusion of signals that cause artifacts and requires manual intervention.

In [Ale+10], the authors design an edge-preserving denoising method for spatial segmentation of peak intensity images ($m/z$-images) derived from MALDI-MSI data. The method utilizes a total variation-based minimization algorithm providing locally adaptive noise reduction. The technique improves the performance of subsequent clustering methods [Ale+10].

Another important aspect is noise within each spectrum due to mass misalignment during data acquisition. In [Bos+19] this misalignment is compensated by exploiting the statistical properties of the characteristic chemical noise background. Thus the mass error is corrected by performing a form of calibration, but the problem of noise due to structurally non-informative spectral data points remains. We consider the aspect of non-informative spectra in Section 3.3.1 in more detail. At this point, it should be briefly stated that these are spectra that do not exhibit the expected characteristic structure of a MALDI-MSI spectrum. Therefore, they do not provide information about the tissue at the corresponding grid position.

In recent years, machine learning algorithms are increasingly used in conjunction with MALDI-MSI. They have a broad range of applications in data analysis, ranging as far as manifold learning using self-organizing maps [VCV20] and targeted feature extraction based on linear methods such as linear discriminative analysis [Cor+19]. The size of MALDI-MSI data poses a problem for effective analysis, therefore another popular usage for machine learning is dimensionality reduction: For example, in [Ing+17] the authors successfully adopted a deep unsupervised neural network to map the 3D data to a 2D manifold. Despite these recent efforts, linear methods in combination with manual feature extraction are considered the standard baseline in MALDI-MSI [Kle+20; Wu+21] due to their simple handling.

In contrast, we apply neural networks directly to raw spectrum data similar to the work of Behrmann et al. in [Beh+17], which we will compare to our work in Section 3.8. In their work, the authors develop a deep convolutional network adapted to the special requirements of the MALDI-MSI data domain. The network is able to handle the characteristics of mass spectrometry data by restricting the local grouping of elements in the convolutional layers. The authors achieve this by estimating the isotope pattern size within the data. This so-called IsotopeNet is highly specialized but was able to outperform a ResNet and LDA classifier on the given data.

In [Mit+21], a 3-layer fully connected and a deep convolutional network achieve higher accuracy than gradient-boosting decision trees. The authors state that combining both networks by averaging their class prediction confidences further improves the prediction.

Lately, more complex models were introduced into MALDI-MSI. In 2021 Arrastia et al. implemented the U-Net architecture to create segmentations of basal cell carcinoma in $m/z$-images [LeC+21]. Since most models in the computer vision community are designed with 2D or 3D images in mind, adapting these architectures is more feasible for processing $m/z$-images than spectral data. On the one hand, $m/z$ images are already in a format that allows common models to be applied directly. On the other hand, the pure spectra are very large and the bundling of multiple spectra to obtain 2D data sets further amplifies this problem.

## 3.3. Model Design

Neural networks provide state-of-the-art performance in medical image analysis tasks such as survival prediction [Yao+17], time series classification [Zhe+14], and image registration [Yan+17; She+19; NKV19]. In contrast, these models are sparsely used in MALDI-MSI. We aim to develop adaptations of these strategies to the MALDI-MSI domain.

Existing applications of neural networks to MALDI-MSI data focus on processing the $m/z$-images [LeC+21], see Figure 3.2. This approach is useful if we are interested in distributions of only few $m/z$-values in larger tissue samples, but is not suited for classification based on features that are more distributed over the $m/z$ spectrum. To the best of our knowledge, the work in [Beh+17] is the first to apply neural networks directly to the spectral data. The proposed architecture is based on residual networks with skip-connections but also incorporates expert knowledge to design the field of view adapted to the spectral patterns prevalent in the data. Therefore, it is not easily transferable to MALDI-MSI with different spectral patterns without expert advice. More recent work [Mit+21] successfully employs 1D convolutional networks, fully connected networks, and a combination of both.

**Model.** In MALDI-MSI the search for *suitable models* has just started. We evaluate multiple architectures including dense fully connected, convolution-based, residual, and recurrent variants. Furthermore, we are experimenting with varying network depths from models consisting of a single layer up to 20 layer deep variants. A common linear layer with softmax activation is used to generate predictions.

One problem of the stochastic batch-wise learning approach in deep learning is the constant change of input distributions. This *covariate shift* causes the layers to constantly adapt to new distributions and slows down training. It has been demonstrated that training converges faster if the input has zero mean and unit variance [LeC+12]. Based on this observation, Ioffe and Szegedy propose a normalization step to adapt the mean $\mu_{\mathcal{B}}$ and variance $\sigma^2_{\mathcal{B}}$ over a given input batch $\mathcal{B}$. The normalization

$$y_i = \gamma \left( \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \right) + \beta \tag{3.1}$$

yields a linear transformation $y_i$ of the input $x_i$ with zero mean and unit variance. Since changing these batch-wise statistics would restrict the layer function, two trainable parameters $\gamma$ and $\beta$ are added. Both allow to change the scale and shift respectively and lift the restrictions on the layer function. In [IS15], the authors demonstrate that *BatchNorm* enables faster convergences during training, serves as a regularizer to the model, and even boosts classification performance for established models on the ImageNet challenge. Therefore, we use *BatchNorm* [IS15] in all model configurations.

Model weights are classically initialized using a uniform distribution over a given interval, the limits are usually chosen in accordance to layer sizes. In [GB10], Glorot and Bengio extensively examine the effects of such initialization on the activations in the model given various activation functions such as sigmoid and hyperbolic tangent. They found that this simple initialization technique causes the variance of the gradient to depend on the layer and it decreases with increasing model depth. As a result, large models tend to have saturated activations (close to zero) and training is slowed down. As a solution Glorot and Bengio introduce a normalization factor to the uniform distribution, changing the limits depending on layer position. This Xavier-Normal distribution [GB10] has established itself as the default weight initialization in deep learning and all model weights in this chapter are initialized following this heuristic.

We implement $L1$-norm

$$\gamma||\Theta||_1 = \gamma \sum_{j=1}^{J} |\theta_j| \tag{3.2}$$

regularization on the layer weights $\Theta$ as part of the loss function with $\gamma$ weighting its influence. This penalizes large weights and discourages overfitting. Neural networks tend to adapt well to training data whenever the feature space of the input samples is large and the training data is sparse at the same time. This is a combination prevalent while working with MALDI-MSI data. Overfitting to the training samples hinders generalization and consequently leads to poor performing classifiers. We employ weight regularization frequently in this work. In those cases, the weighting factor $\gamma$ between the regularizer and data term is provided.

For classification tasks, we use the *categorical-cross-entropy* loss function. Given a set of input vectors $\{x_i\}$ and corresponding targets $\{t_i\}$, the output of our network is defined as $y_i = \mathbf{f}(x_i, \Theta)$. Here $\mathbf{f}$ denotes the network with its parameters $\Theta$, in accordance with (2.1) and (2.2) in Chapter 2. For a binary classification task with two target classes $C_0$ and $C_1$ $y_i$ can be interpreted as the conditional probability $p(C_0|x)$ with $p(C_1|x)$ given by $1 - y_i$. Then the distribution of targets is a Bernoulli distribution

$$\begin{aligned} p(t_i|x_i, \Theta) &= \mathbf{f}(x_i, \Theta)^{t_i}(1 - \mathbf{f}(x, \Theta))^{1-t_i} \\ &= y_i^{t_i}(1 - y_i)^{1-t_i}, \end{aligned} \tag{3.3}$$

where $t = 0$ and $t = 1$ correspond to classes $C_0$ and $C_1$. The cross-entropy is given by the negative log-likelihood

$$L(y, t) = -\sum_{i=1}^{I} \{t_i \ln y_i + (1 - t_i) \ln (1 - y_i)\}, \tag{3.4}$$

which leads to faster training as well as improved generalization for classification tasks compared to the sum-of-squares loss function [SSP+03]. For a multi-class problem with $C$ classes, a general formulation of (3.4) takes the form

$$L(y, t) = -\sum_{i=1}^{I} \sum_{c=1}^{C} \{t_{cn} \ln y_c\}, \tag{3.5}$$

where the true class labels are one-hot encoded — for a 3-class classification problem $t_c \in \{[1,0,0],[0,1,0],[0,0,1]\}$ — and the network outputs $y_c = p(t_c = 1)$ are considered the probabilities for class $c$ [BN06].

The process of **pooling** is a well-known practice in deep learning. It is used to reduce the data dimension and combine information. Different kinds of pooling operations are used, including mean and max pooling. Given a size $n$, the max pooling operation returns the maximum value of the $n$ neighbors of a given data point. Combined with striding larger than one, this merges information from multiple data points into one. In Section 3.7, we apply max pooling with a kernel size of $n = 4$ in conjunction with the Transformer model. A peak in the MALDI-MSI spectrum is associated with a specific chemical structure, for example, a protein or fragment thereof. Peaks consist of four data points building a reappearing pattern, in which the ratio of magnitudes is consistent. Therefore max pooling enables smaller spectrum sizes without too much loss of information.

**Class Balance.** Class balance in medical data sets is often hard to ensure, due to some pathologies being more prevalent. This is especially true for most data sets considered in this thesis. The imbalance can reach a ratio of 4:1, as in the pancreas data set in Section 3.7. We evaluate multiple strategies to ensure class balance:

1. inverse class frequency balancing,
2. class minimum balancing,
3. class maximum balancing.

Inverse class frequency balancing introduces a weighting factor to the loss calculation. We determine the class weights as inverses of the class frequency to increase the influence of classes with few representatives during training. The second and third strategies are preprocessing steps: During the creation of data splits, as described in Section 3.3.1 below, we either include a limited number of samples for each class (**minimum balancing**) or draw repeatedly from classes with few representatives (**maximum balancing**).

**Voting Strategies.** MALDI-MSI data can consist of several thousand spectra per patient tissue sample. In clinical routine, determining class membership for each of those spectra is not always the main interest. Instead, the clinician may only need to know whether the whole tissue sample contains any pathological tissue or none. Consequently, we need to assign class membership to whole patient tissue samples based on the predicted classes of the single spectra therein.

To accommodate this task, we implement two different approaches. Firstly we introduce a **majority voting** strategy to the prediction step. Given a set of class predictions associated with one patient, we assign the majority label to that patient. The result is two-fold. On the one hand, we provide an interpretation of our results more suitable for clinical analysis. On the other, we are able to provide additional confidence scores for whole patient tissue samples calculated from the ratio of class predictions.

As an alternative, we replace the linear classification layer with a custom **voting layer**. We keep the general structure of our network architectures. Single spectra remain as input to the network, but we adopt an online-learning approach as a single batch is used. Each batch consists of a single patient tissue sample. As we know a patient sample is composed of multiple associated spectra. The resulting batch is passed to the network, which processes the spectra individually in the established fashion. The network returns predictions for the individual spectra within the batch (patient sample). Based on these predictions statistics such as the mean, median, minimum, and maximum for the whole batch are calculated. These statistics are then passed to an additional linear layer followed by the softmax activation producing a single prediction for the whole batch (patient sample), which is compared to the patient sample ground truth using the loss function. The error for the patient prediction is then used in the backpropagation step to update the model weights. Keeping the main body of the network the same and only adapting the final classification layer, we enable the usage of whole patient tissue samples as input even though they vary in size.

### 3.3.1. Preprocessing

The workflow of acquiring MALDI-MSI data results in chemical noise originating in tissue preparation and machine-dependent technical noise. Moreover, the produced data is complex and high-dimensional and consequently poses severe challenges for subsequent machine learning methods. We will provide information on established preprocessing methods as well as our own methods developed in conjunction with our classification methods.

**Noise Reduction.** A common strategy to reduce **technical noise** is baseline removal [Nor+07; GRM06] and spectrum normalization with Total Ion Count (TIC), which is the 1-norm applied to each spectrum [Dei+11]. As mentioned in Section 3.1, the baseline is defined as the common shift (increase) in signal intensity. A method for baseline removal is proposed in [Nor+07], where the authors subtract an estimate for the baseline from each spectrum in an iterative manner. The **chemical noise** can cause large differences in peak magnitudes. Consequently, two spectra of similar structures with equal chemical composition can differ in terms of absolute values. This is partially attributed to gradually decreasing ion transmission during acquisition. Without normalization, these artifacts result in inaccurate ion distributions and may lead to incorrect analyses [Dei+11]. We divide each spectrum by its median and apply TIC normalization to compensate for these differences in magnitude so that all spectra are normalized to unit median (Figure 3.3).

An issue arising from the high spatial resolution of MALDI-MSI data is that ground truth — which is typically generated by hand by expert clinicians — is expensive and can often be unreliable on an individual-spectrum basis. As data points with an allocated label may not express the desired morphological structures and thus the measurement might wrongfully be labeled as pathological tissue. We designed a **standard deviation based filter** to identify spectra with little or no information. Spectra with little

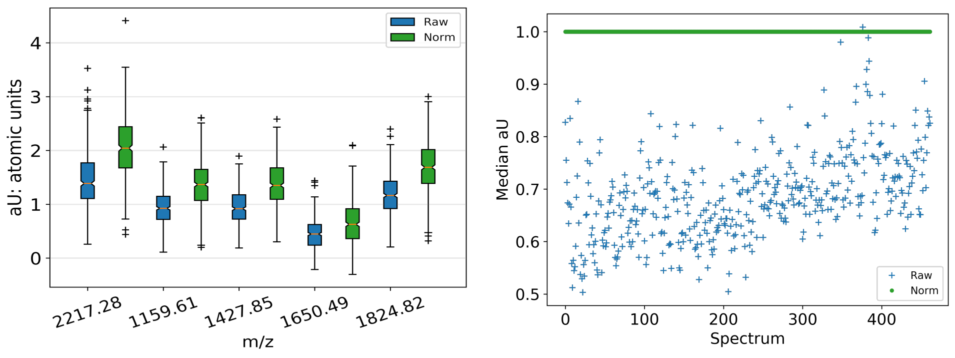
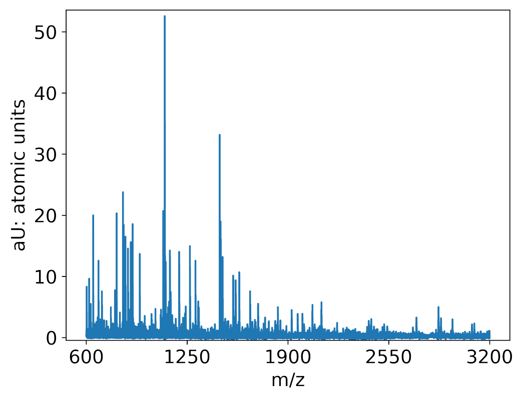


Figure 3.3.: Influence of median filtering on the signal magnitudes of all 462 spectra taken from one patient tissue sample. We multiply a mass spectrum with an intensity-scaling factor to expand the range of the spectra intensities in order to project spectra onto a common intensity scale. **Left**: Effects of median normalization for randomly selected $m/z$ positions. After normalization, the range of peak intensities is increased compared to the raw spectrum. Orange horizontal lines denote medians, boxes show quartiles, and outliers are marked by + symbols. **Right**: Median aU value for all 462 spectra within the patient tissue sample. After normalization, the median value is uniformly set to one.

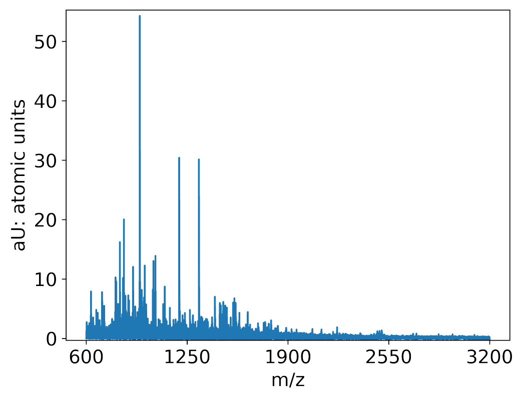
information are associated with low informativeness in the spectral range. As a measure of informativeness, we use the total number of peaks with a magnitude greater than the standard deviation multiplied by a modifiable factor within the spectrum. After expert evaluation of the results for different values, we set this factor to 6. A spectrum is accepted if a predefined threshold is met. For evaluation of the informativeness, the range of data points in each spectrum is restricted to the 60% along the horizontal axis ($m/z$ range). At the end of the spectrum in higher Dalton ranges few peaks are present with little diagnostic value. In Figure 3.4 spectrum examples for different levels of informativeness can be found.

Another possible source of noise is the inconsistent number of **spectra per patient samples**, ranging from as low as two to more than a couple of hundred associated spectra. Samples with too few spectra do not provide a sufficient basis for reliable prediction and may hinder effective training of our learning methods. We consider patient samples with less than 20 spectra as not informative for classification and exclude these samples from evaluation.

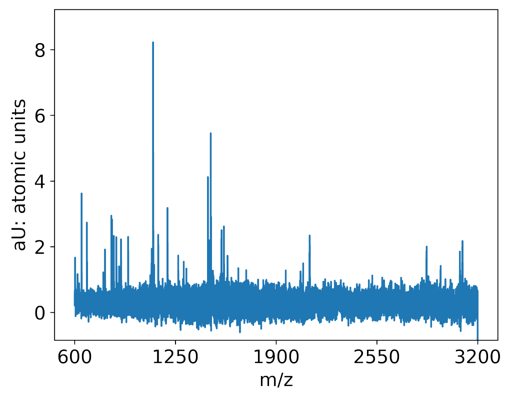
Classifying solely based on single spectra disregards the spatial relationship between neighboring grid points. Pathological tissue typically spreads over multiple of these grid points, thus combining neighboring points could improve robustness. This is implemented by stacking $n$ neighboring spectra of length $m$ together and using a $n \times m$ spectral matrix as input. The spectral matrix incorporates **spatial information** inherent in the tissue sample. We conduct experiments with varying sizes of neighbors $n$.



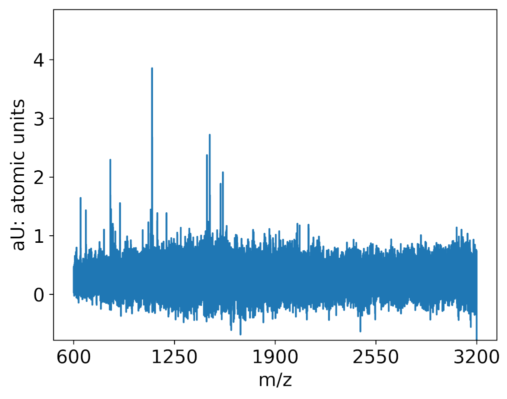
(a)



(b)



(c)



(d)

Figure 3.4.: Examples for spectrum informativeness ratings ranging from an informative spectrum to a spectrum containing only noise. Negative values may occur due to baseline correction during measurement. (**a**) A spectrum is considered good or informative if a threshold of peaks with a magnitude large than the standard deviation within the signal is met. Visually they are characterized by high magnitudes in the first 50% of data along the horizontal axis and diminishing peaks near the end. (**b**) A sufficient spectrum usually displays higher peak concentration in the first 50% of data peaks and a faster decline of peak magnitudes near the end. However, the absolute values of magnitudes remain similar to good spectra. (**c**) A spectrum is considered insufficient if it displays considerably decreased magnitudes (vertical scale) and does not display peak concentration in the first 50% of data. (**d**) Finally, a spectrum contains very little information if it is defined by a noise dominated signal with few peaks and overall small magnitudes.

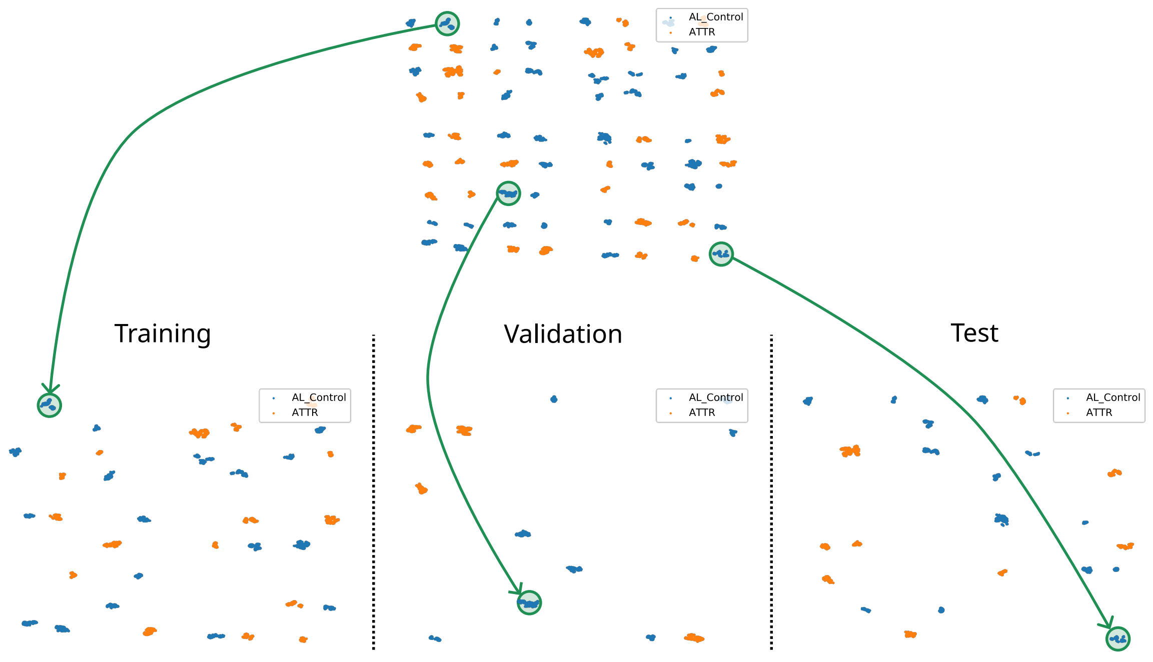


Figure 3.5.: **Top**: Data set creation starts from a full-size TMA. **Bottom**: Patient tissue samples vary in size and are randomly assigned as a whole to one of the three subsets: training (50%), validation (20%), and test (30%). Percentages are computed from spectrum counts. Repeating the process with 3 random seeds yields a three-fold data split used for cross-validation.

**Cross-validation.** In machine learning, *cross-validation* is an important tool for statistical analysis and outlier detection. We apply 3-fold cross-validation for all experiments in this chapter. We first assign each spectrum to a patient. Denoising is applied and patient samples with too few associated spectra are removed. Followed by the creation of three different data splits, each categorized into training-, validation-, and test data subsets. We randomly split a given TMA into these three subsets, an example can be seen in Figure 3.5. Splitting is done for each class individually so that approximately 70% of the data is used for training our algorithms, namely the training data set (50% of total patients' spectra) and the validation set (20% of total). The remaining 30% are used as a test data set to evaluate the final classification performance and therefore unseen during the training phase. We perform the split by selecting full patient tissue samples randomly without replacement and assigning them and all associated spectra to one of the three data sets until the desired share is reached. As a result, the sample distributions in the training, validation, and test data sets vary slightly among the different splits. Repeating this process with varying seeds for the pseudo-random number generator yields the 3-fold cross-validation. We created all data sets in this section using the described assignment process, detailed information about all sets is given in Tables 3.1, 3.7, and 3.13 below.

### 3.3.2. Feature Extraction

The size of raw MALDI-MSI data is relatively large, prohibiting the direct application of most machine learning tools. Manual or algorithmic feature extraction is used to enable the processing of MALDI-MSI spectra. We will introduce some frequently used methods.

Most studies rely on the manual feature extraction by **peak picking**, where a limited number of peaks are chosen and the rest of the spectrum is omitted. The peaks are either identified by an expert or an algorithm. Possible candidates are peaks associated with peptides (proteins) or molecules of interest or peaks with a high distinctiveness measured by a predefined metric. In [Kle+20], the authors propose a typical two-stage approach: First discriminative peptide masses are selected by receiver operating characteristics (ROC) analysis, followed by the computation of the Mann-Whitney-Wilcoxon statistics for each class and peak pair. As a result, a probability is assigned at any given peak for each class indicating whether the spectral intensity at this $m/z$ value is larger than in other classes. The authors restricted the number of selected peaks to 10. Peak picking is able to drastically reduce the data dimension, but leads to a loss of information.

**Principle Component Analysis.** One popular method frequently used in MALDI-MSI to reduce data dimensionality is the *Principle Component Analysis* (PCA). We also consider PCA as a preprocessing step and provide an introduction to it below. We describe how the PCA method creates low-dimensional features from a set of observations $\{x_i\}$. This will allow a better understanding of the method's properties and possible drawbacks when applied to MALDI-MSI. PCA is a feature reduction method that transforms the data such that the direction (component) with the greatest variance is mapped to the first component of a new low-dimensional space of given dimension $p$. We consider a set of observations $\{x_i\}$ with $i = 1, ..., n$, where $x_i \in \mathbb{R}^m$. Note that in most cases we want $p$ to be considerably lower than $m$. Here we consider $p = 1$ and $u_1$ direction vector of that space. Then $u_1^\top x_i$ is the projection of data point $x_i$ to the low-dimensional subspace. Using the set mean $\bar{x}$, the variance of all projections is

$$\frac{1}{n}\sum_{i=1}^{n}(u_1^\top x_i - u_1^\top \bar{x})^2 = u_1^\top S u_1, \; S := \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^\top. \tag{3.6}$$

Here $S$ defines the data covariance matrix. PCA can be interpreted as maximizing the variance in (3.6) with respect to $u_1$ over all vectors of unit length. Enforcing the length constraint using a multiplier yields the Lagrangian

$$L(u_1, \lambda_1) := u_1^\top S u_1 + \lambda_1(1 - u_1^\top u_1). \tag{3.7}$$

Setting $\nabla_{u_1} L$ equal to zero, we see that stationary points are characterized by

$$S u_1 = \lambda_1 u_1, \tag{3.8}$$

i.e., $u_1$ has to be an eigenvector of $S$ with the eigenvalue $\lambda_1$. Multiplying both sides by $u_1^\top$ yields

$$u_1^\top S u_1 = \lambda_1 \; . \tag{3.9}$$

Comparing (3.6) and (3.9), one can see that the variance is largest when $u_1$ is equal to the eigenvector having the largest eigenvalue [BN06]. Repeating this process with additional directions orthogonal to those considered yields further components. Consequently, the data can be represented by $p$ eigenvectors $u_1, ..., u_p$ of the data covariance matrix $S$. This requires us to find $p$ eigenvalues and corresponding eigenvectors, which for large data matrices as in MALDI-MSI is computationally expensive. Nevertheless, PCA is an established tool for data analysis, dimensionality reduction, or feature generation in machine learning and is also frequently applied in the setting of MALDI-MSI.

**Principle Component Pursuit.** In their work, Candès et al. state that although PCA is the most used tool for data analysis and dimensionality reduction, it struggles to perform for noisy data [Can+11]. The authors propose an algorithm for a robust version of PCA (RPCA), aiming for a low-rank matrix $L_0$ recovered from the corrupted measurements. Central to their argument is the assumption that high-dimensional data has a low intrinsic dimensionality, i.e., it can be represented by a lower-dimensional subspace and a sparse "noise" component [EY36; CDS01]: Given the data matrix $M \in \mathbb{R}^{n,m}$, there exists a low-rank approximation of the form

$$M = L_0 + S_0, \tag{3.10}$$

with a low-rank component $L_0$ and sparse noise component $S_0$. The *Principle Component Pursuit* (PCP) algorithm solves

$$\arg \min_{L \in \mathbb{R}^{n,m}, S \in \mathbb{R}^{n,m}} \|L\|_* + \lambda \|S\|_1 \quad \text{s.t.} \quad L + S = M \tag{3.11}$$

to obtain $L_0$ and $S_0$. Here $\|\cdot\|_*$ denotes the nuclear norm and $\|\cdot\|_1$ the $l1$-norm. In the context of MALDI-MSI, the data matrix $M^{n,m}$ consists of stacked spectra $x_i$ with $i \in \{1, ..., n\}$ and the $i$-th row of $L_0$ contains a low dimensional representation of individual spectra $x_i$. The rank of a matrix is the dimension of the vector space spanned by its columns. For low-rank matrices, the number of linearly independent columns is less than the total number of columns, so the corresponding vector space is low-dimensional compared to the vector space of a full-rank matrix. Low-rank representations are a popular method of dimension reduction.

**Non-negative Matrix Factorization.** Another method to calculate a low-rank approximation of a large data matrix $M \in \mathbb{R}^{n,m}$ is *Non-negative Matrix Factorization* (NMF) [LS99]. It has existing applications in the field of compression, feature extraction, and basis learning [VG96; LS00]. Leuschner et al. first introduced this method to MALDI-MSI data in 2019 [Leu+19]. The method determines a number of $p$ characteristic non-negative basis vectors. In the field of MALDI-MSI, it is assumed that a comparatively small number of $p$ structures are represented in the data associated with different proteins or chemical components. The data matrix $M$ consists of $n$ acquisition points each with a spectrum of size $m$. As a result, NMF provides a set of $p$ basis vectors by solving the formulation

$$\arg \min_{K \in \mathbb{R}^{n,p}, X > 0 \in \mathbb{R}^{p,m}} \frac{1}{2} \|M - KX\|_F^2, \tag{3.12}$$

where $K$ is a coefficient matrix and $X$ is a set of stacked spectral basis patterns. In addition, orthogonality constraints on $X$ are introduced such that $XX^\top \approx I$, which results in less correlated pseudo spectra. Input features for subsequent classifiers can be obtained by mapping a spectrum $x_i$ to a feature vector $x_i^f$ with a length of $p$:

$$x_i^f = x_i X^\top \tag{3.13}$$

We test different combinations of feature extraction and classifiers.

## 3.4. Linear Methods

MALDI-MSI is a novel tool for pathological and morphological studies [CC11]. Most recent works concentrate on possible fields of applications and interesting medical use cases [Lon+16]. The classification of the acquired data is an important part of the successful application of MALDI-MSI in the clinical workflow. Despite its crucial function, most studies utilize relatively simple *linear classifiers* [Kle+20; Lon+16] in combination with extensive preprocessing. The feature extraction process serves as dimension reduction and enables the subsequent use of out-of-the-box linear classifiers. This practice is considered state-of-the-art in the MALDI-MSI community. Therefore, we consider the most frequently used variants — Linear Discriminant Analysis (LDA) and Support Vector Machines (SVM) — as benchmarks:

**Linear Discriminant Analysis.** The *Linear Discriminant Analysis* method aims to find a linear combination of features to separate objects into a given number of classes. The classes are modeled as a multivariate normal distribution. The classification of new samples is performed by maximizing the posterior probabilities in regard to the class distributions (Figure 3.6). The class probability density functions are calculated using a subset of the training samples. In this work, we use LDA as implemented in the `scikit-learn` library (version 0.19.1) [Ped+11].

**Support Vector Machine.** In the *Support Vector Machine* method, each sample is represented as an element of a vector space. It can be assumed that vectors resulting from samples of the same class are similar to each other with regard to the decision function. The method aims to find a hyperplane in order to differentiate between samples of different classes. The distance of vectors *closest* to the hyperplane is *maximized* to enable robust classification for unseen samples. The vectors used to calculate the hyperplane are called *support vectors*. In the baseline SVM method, it is assumed that the samples are linearly separable (Figure 3.6). This is not the case for most classification problems with real-world applications such as MALDI-MSI. A nonlinear decision function can be introduced to SVMs using the so-called kernel trick (Figure 3.7). The idea is to nonlinearly embed the given vector space into a higher-dimensional space, in which the samples are linearly separable. A kernel function is used to efficiently solve the resulting maximization problem without having to explicitly compute the embedding

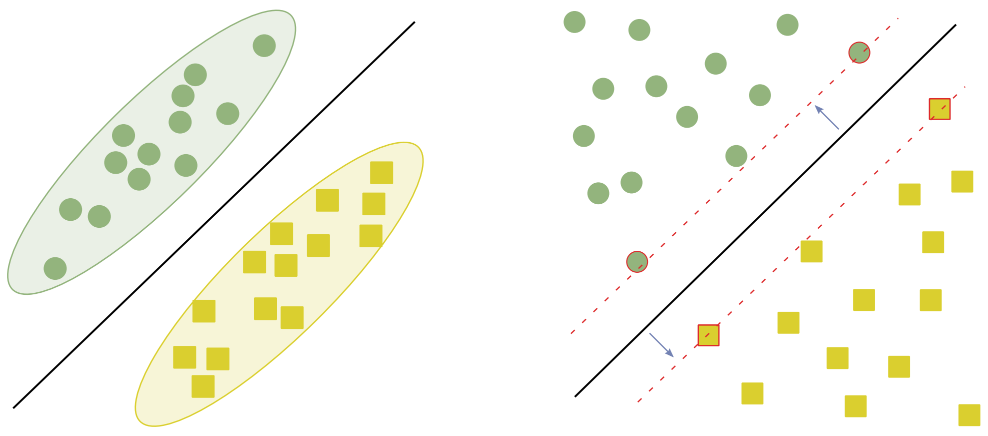


Figure 3.6.: Decision functions for LDA and SVM (linear) methods. **Left**: Distributions for a two-class problem resulting from LDA classification. The black line marks the decision boundary. LDA struggles to model complex class distributions and depends on good feature extraction. **Right**: The SVM chooses the decision boundary by maximizing the margin **(blue arrow)** distance to the closest samples, also called support vectors **(red framed vectors)**. Here an SVM with a linear decision function is shown. Similar to LDA, this method struggles with complex class distributions, which often are not linearly separable (compare Figure 3.7).

into the (potentially very high dimensional) space [SC08]. In addition to the linear SVM, we investigate a radial basis function (RBF) kernel SVM. For multiclass classification, we train binary classifiers in a “one-against-one” strategy and determine the final class decision by a majority voting strategy on the outputs of the pairwise classifiers as implemented in the `scikit-learn` library.

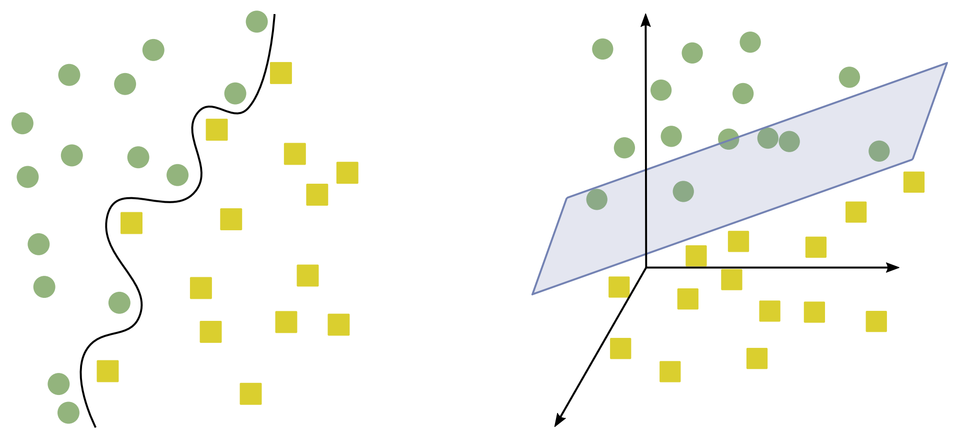


Figure 3.7.: Complex class distributions often require nonlinear decision boundaries. **Left**: Two-class problems with class distributions that are not separable by a linear decision function as used in LDA or SVM (linear). Kernel SVMs are able to model nonlinear decision functions, visualized by the curvy black line. **Right**: The kernel trick enables to efficiently and nonlinearly project the classification problem into a higher dimensional space, in which the decision boundary becomes a hyperplane **(blue rectangle)**.

## 3.5. Ovarian Cancer

In the following, we present the results for the classification of ovarian cancer subtypes. A focus is on the search for a suitable network architecture as well as the evaluation of the models when combined with established feature extraction techniques.

### 3.5.1. Data

Epithelial ovarian cancer (EOC) is an inhomogeneous disease with multiple histological subtypes. Five main subtypes have been described by the World Health Organization with diverse molecular structures, clinical behavior, and therapeutic prognoses [Mei+16; PDE18]. Response to standard therapy can differ vastly and clinical outcomes could be improved with adapted therapy approaches for each EOC subtype. Serous Ovarian cancer (OC) tumors are the most frequent EOC subtypes (70%) and are of particular clinical interest. High-grade serous ovarian cancer (HGSOC) and low-grade serous ovarian cancer (LGSOC) differ on the molecular level, containing mutations on different genes, such as BRAF or BRCA, resulting in genetic instability. The precursors of LGSOC are serous borderline tumors (sBOT), these two types are considered to be difficult to distinguish due to their similar morphological patterns.

Ovarian clear cell carcinoma (OCCC) is a rare EOC subtype characterized by clear cytoplasm and endometriotic foci, which are able to evolve to cancer [MSK16]. The described EOC subtypes require customized chemotherapeutic and targeted therapies. The classification of these subtypes remains challenging. There frequently are areas with morphological overlap between the HGSOC and LGSOC subtypes and OCCC structures

Table 3.1.: Data set sizes after random split (percentage of total in parentheses). Samples from individual patients are distributed across the three data sets. Care is taken to ensure that approximately 50% of the patients are used as training data, 20% are used for evaluation during training, and 30% are used in the test data set. The training and test data sets are preferentially populated, so the evaluation data set can be smaller.

| | Training | | Validation | | Testing | |
|---|---|---|---|---|---|---|
| | Patient | Spectra | Patient | Spectra | Patient | Spectra |
| Data set I | 56 (53.3%) | 12051 (52.4%) | 16 (15.2%) | 3470 (15.1%) | 33 (31.4%) | 7483 (32.5%) |
| Data set II | 56 (53.3%) | 11749 (51.1%) | 20 (19.0%) | 3743 (16.3%) | 29 (27.6%) | 7512 (32.7%) |
| Data set III | 54 (51.4%) | 12059 (52.4%) | 18 (17.1%) | 3584 (15.6%) | 33 (31.4%) | 7361 (32.0%) |

can be included in HGSOC.

In this study, our data set contains tissue specimens from 67 patients with a total of 111 patient tissue samples, of which 8 are rejected due to the low spectra count within. The samples are distributed to the EOC subtypes as follows: sBOT (19 tissue samples/ 14 patients), HGSOC (31 tissue samples/ 19 patients), LGSOC (26 tissue samples/ 14 patients), OCCC (35 tissue samples/ 20 patients). Each spectrum consists of 8668 values.

The study was designed by Silvia Darb-Esfahani and Carste Denkert from the Institute of Pathology, Charité-Universitätsmedizin Berlin. The annotation was performed by Eliane T. Taube from the Institute of Pathology, Charité-Universitätsmedizin Berlin.

### 3.5.2. Experiments and Results

First, we focus on finding a suitable network architecture to process MALDI-MSI data. We present the results for different architectures including the *IsotopeNet* proposed in [Beh+17], followed by a combination of feature extraction described in Section 3.3.1 and our architectures. Finally, we compare the best-performing model to the commonly-used baseline classifier methods LDA and SVM.

**Architecture.** We train on data set II (Table 3.1) in order to evaluate the various architectures. Hyperparameters are chosen by random search: Batch size [50, 400], epochs [50, 200], learning rate [$10^{-2}$, $10^{-6}$], weight regularization [$10^{-1}$, $10^{-4}$], kernel size [128, 256], and feature channels [4, 8], see Table 3.2. All models are trained on an NVIDIA RTX 2080 GPU with 8 GB memory. In general, all models are $L1$-regularized with weights ranging from $10^{-3}$ to $10^{-5}$ in order to avoid overfitting, given the limited amount of data and large feature size.

Models with a depth of more than two layers perform worse than relatively shallow few-layer architectures. Particularly interesting is the poor performance of IsotopeNet, which indicates that the specialized filter design is not readily applicable to our MALDI-MSI data. The best-performing architectures are a single layer fully connected network as well as a two-layer convolutional network with 86% and 89% of patient samples

Table 3.2.: Parameter configurations and prediction accuracy on the test set for spectra (values for patient samples in parentheses) using different neural network architectures. Best accuracy marked in bold numbers.

| Model | Accuracy | $L1$ weight | Batch Size | LR | Kernel Size | Feature Channel |
|---|---|---|---|---|---|---|
| Fully connected | 0.7661 (0.8621) | $10^{-3}$ | 150 | $10^{-3}$ | - | - |
| CNN 1 | 0.7866 (**0.8965**) | $10^{-4}$ | 150 | $10^{-3}$ | 128 | 4 |
| CNN 2 | **0.8073** (0.8333) | $10^{-4}$ | 200 | $10^{-4}$ | (256, 128) | (8,8) |
| IsotopeNet [Beh+17] | 0.5482 (-) | $10^{-4}$ | 150 | $10^{-4}$ | 3 | 8 |

correctly classified. Both models achieve an increase of 25% compared to the IsotopeNet. Consequently, we test both architectures in combination with feature extraction methods from Section 3.3.1, which are commonly applied in MALDI-MSI.

**Feature Extraction.** Starting from our findings during the model search, we continue to test the most promising candidates *fully connected* and *CNN 2* with *feature extraction* methods. Generally, neural networks do not need manual or algorithmic feature extraction, since the first few layers in a model are basically designed to perform this task. Nevertheless, passing processed features into a network can be beneficial, e.g., for dimensionality reduction. In combination with linear methods, the NMF method proved to be efficient when applied to MALDI-MSI data [Leu+19]. When we use NMF to generate features for our neural networks, we only achieve accuracies of up to 50%. This is far less than when we apply the networks directly to the raw spectra, with accuracies of up to 80%. For most parameter configurations, neither the fully connected nor the CNN model can achieve reasonable accuracy. The results are often in the range of 25% and indicate that no classification could be achieved when NMF is used as a feature extractor.

The conceptually related PCA and PCP methods achieve promising results in combination with both architectures. But the gain in accuracy does not justify the application of these time and memory consuming algorithms. The PCP method with 50 features takes approximately 19 hours of computing time and uses up to 11 GB of memory. The PCA takes considerably less time to compute, under a minute for 512 features, but does not provide the same level of accuracy on the patient tissue level. Both methods utilize a matrix representation of the spectral data and perform a variant of singular value decomposition (SVD) favoring smaller matrices. Therefore, it is expected that computational time and storage requirements will increase for data sets with larger spectra. The spectra evaluated here are relatively small, with a length of 8668. We are confident that a more thorough parameter tuning achieves higher accuracies and the use of feature extraction methods does not add value compared to applying the models to the raw spectra.

Finally, we test the grouping of multiple neighboring pixels into one sample. We are limited by the length of our spectral data and memory constraints on the hardware side. The best results are obtained using a five spectra neighborhood and the convolutional

Table 3.3.: Prediction accuracy for spectra (values for patient tissue samples in parentheses). The best-performing models of the selection process are fully connected and CNN1. We evaluate performance when combining the models with NMF, PCA, and PCP. The number of features is given in the feature size column. Our neighbors' approach increases the input size by the number of neighbors used, provided in the feature size column. The PCP and PCA methods allow similar model performance compared to using the full spectral range if a sufficient number of components are used. The NMF and Neighbor methods do not improve model performance. Best accuracy marked in bold numbers.

| | Fully connected | | | CNN 1 | |
|---|---|---|---|---|---|
| Method | Feature Size | Accuracy | Method | Feature Size | Accuracy |
| NMF | 60 | 0.3991 (0.3) | NMF | 256 | 0.4217 (0.3793) |
| | 1600 | 0.4293 (0.3) | | 1200 | 0.3911 (0.3) |
| PCA | 128 | 0.6097 (-) | PCA | 512 | 0.7558 (0.8181) |
| | 512 | **0.8196** (0.8965) | | | |
| | | | PCP | 1 | 0.7571 (0.8275) |
| | | | | 15 | 0.7892 (0.8965) |
| | | | | 50 | 0.7814 (**0.9310**) |
| | | | Neighbors | 5 | 0.7328 (0.7741) |

architecture. Classification accuracy does not improve using grouped spectra, therefore we omit the adaptation of established preprocessing into our classification pipeline. The inherent feature extraction within the convolutional network outperforms all methods except the memory-intensive PCP (11 GB). The gain in classification accuracy on the patient level comes at the cost of a loss on the spectral level. Table 3.3 contains the most important results for our feature extraction experiments.

**Ovarian Cancer Classification.** In the following, we present our work *MALDI-Imaging for Classification of Epithelial Ovarian Cancer Histotypes from a Tissue Microarray Using Machine Learning Methods* [Kle+19] in **Proteomics–Clinical Applications** 2018. We use the following preprocessing steps:

- Median normalization
- Removal of tissue samples with small tumor regions
- Majority voting (tissue sample — patient — accuracy)
- 3-fold cross-validation (see Table 3.1)

As baseline methods, we employ the LDA and SVM methods. In addition to the linear variant of SVM (SVM-lin) we also test a radial basis function (RBF) kernel SVM (SVM-rbf), which could solve more complex classification problems. Based on the results from

Table 3.4.: Prediction accuracy on the test data sets for spectra (values for patients in parentheses) using different classification algorithms. Best accuracy marked in bold numbers.

| | LDA | SVM-lin | SVM-rbf | FCN | CNN |
|---|---|---|---|---|---|
| Data set I | 0.61 (0.82) | **0.77** (**0.85**) | 0.76 (0.79) | 0.76 (0.82) | 0.76 (0.79) |
| Data set II | 0.58 (0.79) | 0.77 (0.79) | 0.68 (0.72) | 0.77 (0.86) | **0.82** (**0.97**) |
| Data set III | 0.57 (0.79) | **0.72** (0.76) | 0.70 (0.70) | 0.69 (**0.82**) | 0.70 (0.79) |
| ∅ | 0.59 (0.80) | 0.75 (0.80) | 0.71 (0.83) | 0.74 (0.83) | **0.76** (**0.85**) |

our preceding experiments we choose to train a fully connected network (FCN) with a single layer and a two-layer network (CNN) with one convolutional layer (four feature maps, kernel size 120, stride 1). The second layer in the convolutional model is again a fully connected layer. Both model weights are $L1$-regularized to avoid overfitting with weights of $10^{-3}$ for the FCN architecture and $10^{-4}$ for the convolutional network. Weights are randomly initialized as stated in Section 3.3 and training is performed using the Adam optimizer with a learning rate of $10^{-3}$, a reduce-on-plateau reduction strategy as well as a batch size of 150. Multiple parameter configurations are tested using random search, among which the setting in (Table 3.2) yields the best results.

The linear discriminant analysis yields the least accurate results. On average, it achieves an accuracy of 59% on the spectra level and 80% on the patient level. The linear SVM (75% individual spectra, 80% patient) performs similarly to the linear neural network (74% individual spectra, 83% patient). SVM-rbf yields an accuracy of 71% on the spectra level and 74% on the patient level after voting. The CNN achieves the best performance with an accuracy of 76% and 80% on the spectra and patient level respectively.

Overall, the differences in accuracy for the top 3 models vary only slightly (Table 3.4). Figure 3.8 displays the predicted subtypes on the patient level for the SVM-lin and CNN methods. Regarding specificity on the patient level, all methods (LDA, SVM-lin, SVM-rbf, FCN, CNN) achieve above 91% on OCCC, sBOT, and LGSOC, and 72-93% on HGSOC (see Table 3.6).

On the individual spectrum level, the specificity ranges from 84% (OCCC, sBOT, LGSOC) to 75-87% (HGSOC). Regarding sensitivity the network models and SVM-lin achieve the best results, scoring 80-100% on the patient and 75-89% on the spectra level (OCCC, sBOT, HGSOC). The LGSOC histotype displays the lowest sensitivity values of all subtypes (64-75% on patient, 58-59% on spectra). We are able to mostly classify the OCCC, HGSOC, and sBOT subtypes correctly. In contrast, LGSOC classification accuracy is considerably lower, falling below 50% for all methods. Confusion matrices for all methods indicate that LGSOC is mainly confused with OCCC.

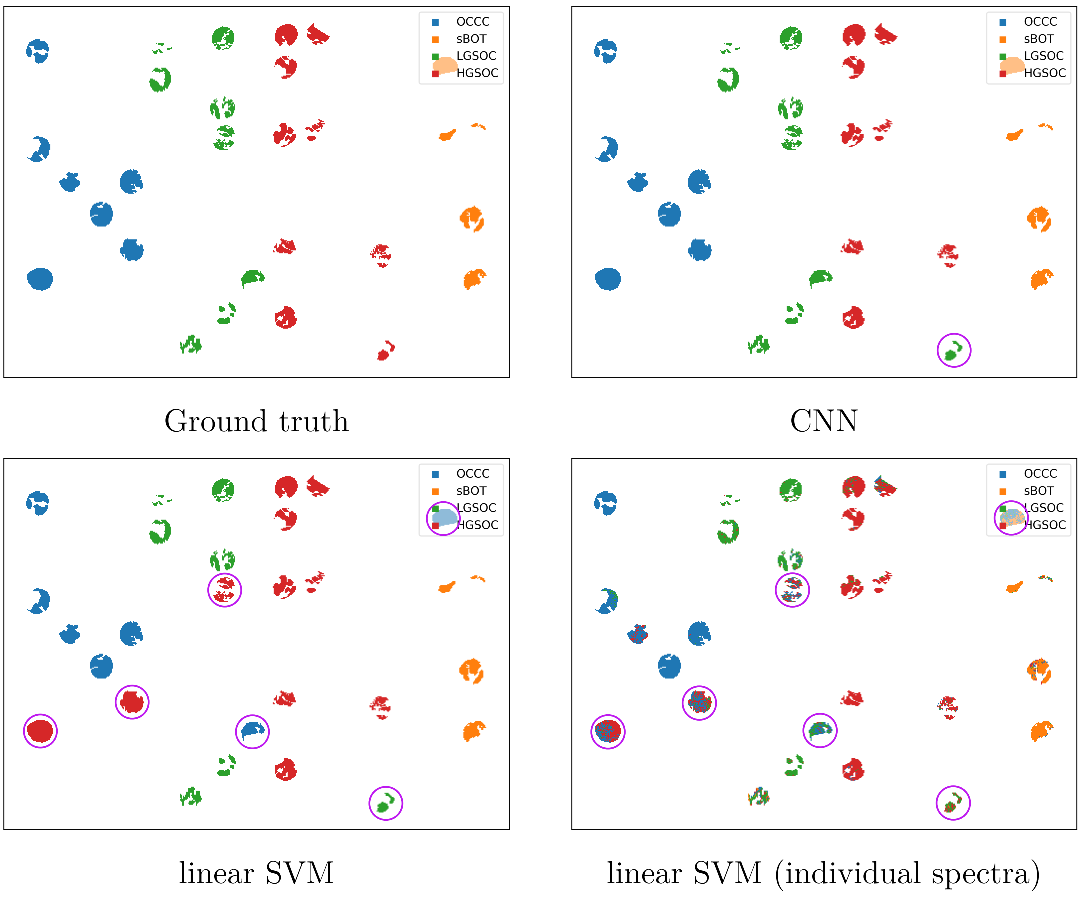


Figure 3.8.: EOC histotype prediction from TMA (data set II). **Top left**: Ground truth test set. **Top right**: CNN results for patient prediction with a sensitivity of 69-100% and a specificity of 90-99%. **Bottom left**: SVM-lin results for patient prediction with a sensitivity of 65-93% and a specificity of 87-99%. **Bottom right**: SVM-lin results in sensitivity of 59-83% and specificity of 86-98% on the individual spectrum level. Incorrectly classified patient samples are marked in purple (encircled).

Table 3.5.: Sensitivity on the best test data set for spectra (values for patient in parentheses) using different classification algorithms. Values are provided per EOC histotype and averaged over three different splits. Best sensitivity marked in bold numbers.

| Histotype | LDA | SVM-lin | SVM-rbf | FCN | CNN |
|---|---|---|---|---|---|
| OCCC | 0.67 (0.93) | 0.80 (0.80) | 0.77 (0.79) | 0.79 (0.85) | **0.85** (**0.97**) |
| sBOT | 0.49 (0.60) | 0.75 (0.93) | 0.56 (0.60) | 0.78 (0.93) | **0.80** (**1.00**) |
| LGSOC | 0.53 (0.71) | **0.59** (0.64) | 0.49 (0.56) | 0.58 (**0.75**) | **0.59** (0.69) |
| HGSOC | 0.59 (0.89) | 0.83 (0.89) | **0.89** (**0.93**) | 0.78 (0.86) | 0.76 (0.82) |

Table 3.6.: Specificity on the best test data set for spectra (values for patient samples in parentheses) using different classification algorithms. Values are provided per EOC histotype and averaged over three different splits. Best specificity marked in bold numbers.

| Histotype | LDA | SVM-lin | SVM-rbf | FCN | CNN |
|---|---|---|---|---|---|
| OCCC | 0.85 (0.92) | **0.91** (0.94) | 0.92 (0.96) | **0.91** (0.94) | 0.90 (**0.97**) |
| sBOT | 0.95 (**0.99**) | **0.98** (**0.99**) | **0.98** (**0.99**) | 0.97 (**0.99**) | **0.98** (**0.99**) |
| LGSOC | 0.84 (0.94) | 0.92 (0.93) | **0.95** (**0.97**) | 0.91 (0.91) | 0.92 (0.93) |
| HGSOC | 0.79 (0.88) | 0.86 (0.87) | 0.75 (0.72) | 0.84 (0.93) | **0.87** (**0.90**) |

### 3.5.3. Summary and Discussion

We applied neural networks to MALDI-MSI data to distinguish between EOC subtypes. We proposed model architectures to process the given data on an individual spectrum level and provided class membership predictions for whole patient samples, usable in clinical routine. Different dimensionality reduction and feature extraction algorithms were tested and finally, our networks' performances were compared to the established linear classification methods of LDA and SVM. Our neural network-based pipeline achieved the highest sensitivity and specificity scores. These results encourage further investigation of such classifiers in the context of MALDI-MSI.

For the first time, we demonstrated that MALDI-MSI combined with machine learning approaches can classify different histologic subtypes of epithelial ovarian cancer. Most mass spectrometry imaging studies attempt to classify only tumor types that have major histological differences, for example, squamous and adenocarcinoma [Med+12; AW15; Kri+16], or tumors in different organs. The differentiation of EOC subtypes is a much more detailed question and of greater interest in routine pathology, as the morphological structures of EOC subtypes can be quite similar. [Mei+16].

The best overall EOC histotype prediction was achieved by using a two-layer CNN classifier. We attribute the shallow model design to the relatively small number of qualitatively different tissue samples in this data set, the high feature dimension, and the consequential susceptibility to overfitting. CNN, FCN, and SVM-lin were all able to distinguish between the three EOC types OCCC, sBOT, and HGSOC resulting in sensitivity values from 82% to 100% after voting. The classification of LGSOC tissue appears to be more challenging. Just recently LGSOC and HGSOC had been described as distinct and autologous histotypes and morphological structures are highly similar.

Additionally, compared to other groups, the number of samples in the data set is low. Due to limitations in the accuracy of ground truth labels, small stromal regions may be present in the tumor regions. Hence, classification based on individual spectra results in decreased values for accuracy, sensitivity, and specificity.

Subsequent studies will have to investigate, based on a larger cohort, what impact technical variables could have on the classification robustness. Furthermore, the model can be adapted further to address the challenges posed by MALDI-MSI data.

## 3.6. Amyloidosis

Building on the results from the ovarian cancer data set, we apply the network classifiers to differentiate cardiac amyloidosis plague. We use existing models as a baseline to further investigate architectural designs for neural networks in MALDI-MSI, particularly emphasizing the deployment of residual connections and recurrent models.

### 3.6.1. Data

Amyloidosis is caused by the extracellular deposition of various proteins. Up to a hundred different proteins have been identified as causative agents for amyloidosis. Systemic cardiac amyloidoses (CA) are an uncommon, but underdiagnosed cause of heart failure. Understanding of underlying ATTR amyloidosis mechanisms can be improved and therapy options developed, if ATTR is reliably detected and distinguished from secondary types of amyloidosis, such as amyloid light-chain (AL). Therefore, simultaneous observation of cardiac amyloidosis and its secondary effect on the adjacent tissue is one of the key requirements for individual therapy and diagnostic developments.

The amyloidosis data set contains tissue samples from 73 patients. During preprocessing, four patient tissue samples were rendered insufficient due to containing too few associated spectra. These samples were excluded, resulting in a total of 69 patient samples. The samples are distributed as follows: ATTR (33 tissue samples/ 33 patients), of which 11 tissue samples also contain inflammation, AL (32 tissue samples/ 32 patients), and 8 tissue samples/patients which do not contain amyloidosis. As we are interested in the identification of ATTR, two groups are defined. The first class contains solely ATTR and the second class combines the AL and healthy samples (AL-Control). Each spectrum consists of 8666 $m/z$ positions. Again, we create three different data splits for cross-validation (Table 3.7)

The study was designed by Carsten Tschöpe and Sophie van Linthout from the Berlin-Brandenburg Center for Regenerative Therapies and Berlin Institute of Health Center for Regenerative Therapies (BCRT), Charité. The annotation was performed by Karin Klingel from the Institute for Pathology and Neuropathology, University Hospital Tübingen. Founding was provided by the Pfizer Aspire grand.

### 3.6.2. Experiments and Results

Building on our experiments in Section 3.5, we apply the network models to amyloidosis data. For the ovarian cancer data set, the most promising feature extraction is PCA; consequently, we also test if feature extraction provides a benefit for the amyloidosis data set. In addition to the convolutional and fully connected model, we also employ residual as well as recurrent network architectures, which have been shown to achieve good results on time-series and 1D signal classification [GMH13; LQH16]. All experiments are again conducted on an NVIDIA RTX 2080 GPU with 8 GB memory. Starting from the findings on the ovarian cancer sub-typing problem, the best-performing models are tested on the amyloidosis data set. We keep the model architectures but choose a new set of

Table 3.7.: Data set sizes after random split (percentage of total in parentheses). Samples from individual patients are distributed across the three data sets. Care is taken to ensure that approximately 50% of the patients are used as training data, 20% are used for evaluation during training, and 30% are used in the test data set. The training and test data sets are preferentially populated, so the evaluation data set can be smaller.

| | Training | | Validation | | Testing | |
|---|---|---|---|---|---|---|
| | Patient | Spectra | Patient | Spectra | Patient | Spectra |
| Data set I | 34 (49.3%) | 72534 (51.1%) | 12 (17.4%) | 24094 (17.0%) | 23 (33.3%) | 45236 (31.9%) |
| Data set II | 36 (52.2%) | 74619 (52.6%) | 11 (15.7%) | 22245 (15.7%) | 22 (31.9%) | 45000 (31.7%) |
| Data set III | 35 (50.7%) | 71392 (50.3%) | 14 (20.3%) | 26920 (19.0%) | 20 (29.0%) | 43552 (30.7%) |

hyperparameters by random search: Batch size [50, 150], epochs [10, 50], learning rate [$10^{-3}$, $10^{-5}$], weight regularization [$10^{-3}$, $10^{-4}$], kernel size [4, 8, 128, 256], and feature channels [4, 8]. For the CNN model, the best parameter settings regarding accuracy are a kernel size of 8 with 4 channels, weight regularization with the factor $10^{-4}$, a learning rate of $10^{-5}$, and a rate decay of 0.9 in combination with a reduce-on-plateau schedule. For the FCN model, the learning rate starts at $10^{-3}$ and the rate scheduling is the same. The weight regularization factor is $10^{-3}$.

**Feature Extraction.** Again, feature extraction does not lead to satisfying results. For the two-class problem, NMF with varying numbers of basis patterns results in an accuracy of 59%, suggesting random assignment of classes. The sensitivity and specificity scores of 0.28 and 0.94 show however that the model indeed does not assign the classes at random but rather that one class dominates the predictions, resulting in a prediction accuracy reflecting class distributions.

Feature extraction seems to be beneficial only in the combination of PCA with a small number of components and the FCN model. This combination achieves a prediction accuracy of 81% on the patient tissue sample level and 75% on the spectral level with reasonable sensitivity and specificity levels of 0.77 and 0.73 (see Table 3.8). Interestingly, feature extraction worsens the performance of the CNN model, even resulting in predictions worse than chance. The CNN model with PCA feature extraction tends to emphasize the AL-Control class and regularly misclassifies the dominant ATTR class. This indicates poor model generalization and emphasizes that the CNN model is not able to learn reasonable class distributions based on the features created by the PCA algorithm. Similar to the results in Section 3.5 for the ovarian cancer data set, feature extraction seems to limit the models' capabilities and is only effective in combination with the FCN model without improving results. Therefore, we omit the feature extraction in the following.

**Amyloidosis Plaque Classification.** Residual connections allow deep networks to achieve higher accuracies (Section 2.1). So far our models have been very shallow, but

Table 3.8.: Results of amyloidosis classification for best-performing models in the ovarian cancer setting. An additional feature extraction step hinders conversion for the CNN model but achieves reasonable results in terms of prediction accuracy for the FCN model. NMF does not work in combination with a dense model and leads to increased misclassification on both the spectrum and patient level. Based on these results and our observations from the ovarian data set, we omit the evaluation of NMF in combination with CNN models. Overall we constitute that feature extraction does not provide significant benefits for the prediction of amyloidosis plaque types. The sensitivity and specificity values are calculated on the spectral level. Best results marked in bold numbers.

| Model | Feature Extraction | Accuracy (spectrum) | Accuracy (patient) | Sensitivity | Specificity |
|---|---|---|---|---|---|
| FCN | NMF (64) | 0.5995 | 0.5909 | 0.2882 | 0.9406 |
| FCN | PCA (16) | 0.7556 | 0.8181 | **0.7731** | 0.7364 |
| FCN | None | 0.7556 | **0.8182** | **0.7731** | 0.7364 |
| CNN | PCA (16) | 0.4074 | 0.4091 | 0.4656 | 0.3437 |
| CNN | None | **0.7624** | 0.7273 | 0.7510 | **0.7750** |

for our larger models, we are evaluating whether the residual connections can improve the models' predictions. For the ovarian cancer data set, FCN and CNN achieve the best performances, followed by a linear SVM. Therefore, we apply these models again to the task of amyloidosis subtyping and also test new architectures: a two-layer LSTM and a two-layer and four-layer convolutional models with residual connections. Again, we use the commonly used LDA and SVM methods as a baseline to evaluate the performance of our networks.

We train all classifiers on individual spectra while monitoring the performance on the validation set. The model with the highest accuracy score on the validation data is used to evaluate the performance on the unseen test data. We retain the processing pipeline established in Section 3.5, consisting of median normalization, removal of patient samples with small tumor regions, 3-fold cross-validation, and a majority voting scheme.

In the first experiment, we focus on the comparison of neural network based classifiers and the baseline LDA and SVM. Model configurations yielding the best accuracy for four different model architectures are presented in Table 3.9. Two models are single-layer networks (FCN, CNN), and the Recurrent as well as the Residual model consists of two layers. As explained in the model search section 3.5, parameter configurations are determined using random search. As before, increasing the depth of model architectures leads to improved training accuracy up to 99% for the Recurrent model, but prediction accuracy on the validation set drops.

Following the 3-fold cross-validation scheme, we first compare the network model performances to LDA and SVM. The comparison is carried out on the first split and all results are listed in Table 3.10. The simple FCN and LDA underperform with a patient accuracy of 65% and 69%. From the experiments in Table 3.8, we know that the FCN model does not benefit from features previously extracted by other methods. For NMF

Table 3.9.: Parameter configurations for models with the highest accuracy on the test set. The most impactful parameters are the learning rate and regularization factor. Two simple-layer models (FCN, CNN) and one two-layer model (Residual, Recurrent) were tested.

| Model | $L1$ weight | Batch Size | LR | Kernel Size | Channels |
|---|---|---|---|---|---|
| FCN | $10^{-3}$ | 100 | $10^{-4}$ | - | - |
| CNN | $10^{-4}$ | 150 | $10^{-5}$ | 128 | 4 |
| Residual | $10^{-3}$ | 50 | $5 \cdot 10^{-3}$ | (128, 128) | (8,8) |
| Recurrent | $10^{-4}$ | 100 | $10^{-5}$ | (256, 128) | 4 |

Table 3.10.: Prediction accuracies for all network classifiers and both baseline models on the first cross-validation split. Except for the dense FCN model, all networks achieve promising results with an accuracy ranging from 78% to 86% on the patient level. The best prediction of baseline methods is achieved by the linear SVM with 78%. The networks outperform the baseline methods by 8% on the patient level. Best results marked in bold numbers.

| Model | Split | Accuracy (spectrum) | Accuracy (patient) | Sensitivity | Specificity |
|---|---|---|---|---|---|
| FCN | I | 0.5939 | 0.6522 | 0.5369 | 0.6592 |
| Residual | I | **0.8334** | 0.7826 | 0.8267 | **0.8411** |
| CNN | I | 0.8048 | **0.8696** | **0.8414** | 0.7627 |
| Recurrent | I | 0.8165 | 0.8261 | 0.8146 | 0.8188 |
| LDA | I | 0.6594 | 0.6957 | 0.5964 | 0.7315 |
| SVM-lin | I | 0.7951 | 0.7826 | 0.8124 | 0.7754 |
| SVM-rbf | I | 0.7398 | 0.7391 | 0.7570 | 0.7201 |

the results get worse and in combination with PCA the model achieves the same level of accuracy as used without it. CNN models achieve better results without prior feature extraction, probably due to the small size of the selected features. We decided not to use PCP, because previously the time taken to calculate was too long and the benefit minor.

Both SVM variants display improved results compared to LDA and FCN with patient accuracy of 78% and 73%. The linear SVM variant interestingly outperforms its RBF-kernel counterpart in every metric, delivering more reliable predictions on both the spectra and patient level. But both variants lack the performance of the remaining three network models with a gap in accuracy ranging from 4% to 13%.

These results suggest focusing on the network models, as the conventional machine learning tools under-perform and do not provide any benefits over networks. To determine the overall best-fitting model type, we evaluate all network models on the full cross-validation split. In Table 3.11, the accuracies on spectrum and patient level as well as the sensitivity and specificity of patient prediction are listed. The Residual model achieves the best accuracy of all models when averaging over all splits. On the spectrum level,

Table 3.11.: Prediction accuracies for all network classifiers. Displayed are the remaining two splits of the 3-fold cross-validation and the mean accuracies over all splits including the results shown in Table 3.10. Best results marked in bold numbers.

| Model | Split | Accuracy (spectrum) | Accuracy (patient) | Sensitivity | Specificity |
|---|---|---|---|---|---|
| FCN | I | 0.5939 | 0.6522 | 0.5369 | 0.6592 |
| | II | 0.7556 | 0.8182 | 0.7731 | 0.7364 |
| | III | 0.7259 | 0.7 | 0.8472 | 0.5912 |
| | ∅ | 0.6918 | 0.7235 | 0.7191 | 0.6623 |
| Residual | I | **0.8334** | 0.7826 | 0.8267 | **0.8411** |
| | II | 0.8266 | 0.7727 | 0.7397 | 0.9218 |
| | III | 0.7539 | 0.75 | 0.8830 | 0.6106 |
| | ∅ | 0.8046 | 0.7684 | 0.8165 | 0.7912 |
| CNN | I | 0.8048 | **0.8696** | **0.8414** | 0.7627 |
| | II | 0.7624 | 0.7273 | 0.7510 | 0.7750 |
| | III | 0.7478 | 0.7 | 0.8623 | 0.6207 |
| | ∅ | 0.7717 | 0.7656 | **0.8182** | 0.7195 |
| Recurrent | I | 0.8165 | 0.8261 | 0.8146 | 0.8188 |
| | II | 0.7614 | 0.6818 | 0.6472 | 0.8866 |
| | III | 0.7545 | 0.8 | 0.7801 | 0.7261 |
| | ∅ | 0.7775 | 0.7693 | 0.7473 | 0.8105 |

it correctly classifies 80% of samples and 75% on the patient level. Only the Recurrent model displays a better patient level prediction accuracy of 76%.

In Figure 3.9, the patient prediction for the Recurrent and Residual model as well as the best-performing SVM on the evaluation split one are provided. On the patient level, the Recurrent model achieves the overall best accuracy followed by the Residual model, which achieves better performances on the spectra level.

### 3.6.3. Leaking Information

A major problem in the combination of MALDI-MSI and machine learning tools is often insufficient separation of training and test data. As pointed out in [Des22], most studies rely on manual or algorithmic feature extraction prior to applying machine learning algorithms, which eventually leaks information from the test set into the training set. Many researchers in the field of MALDI-MSI do not have a machine learning background and are unaware of this problem.

To verify that the given models learn meaningful features for the classification task, we applied a *random label test* for our best-performing model. This approach is similar to the suggestions in [Des22] to test whether information leaked into the test set. In this test, the class labels are randomly assigned to each patient.

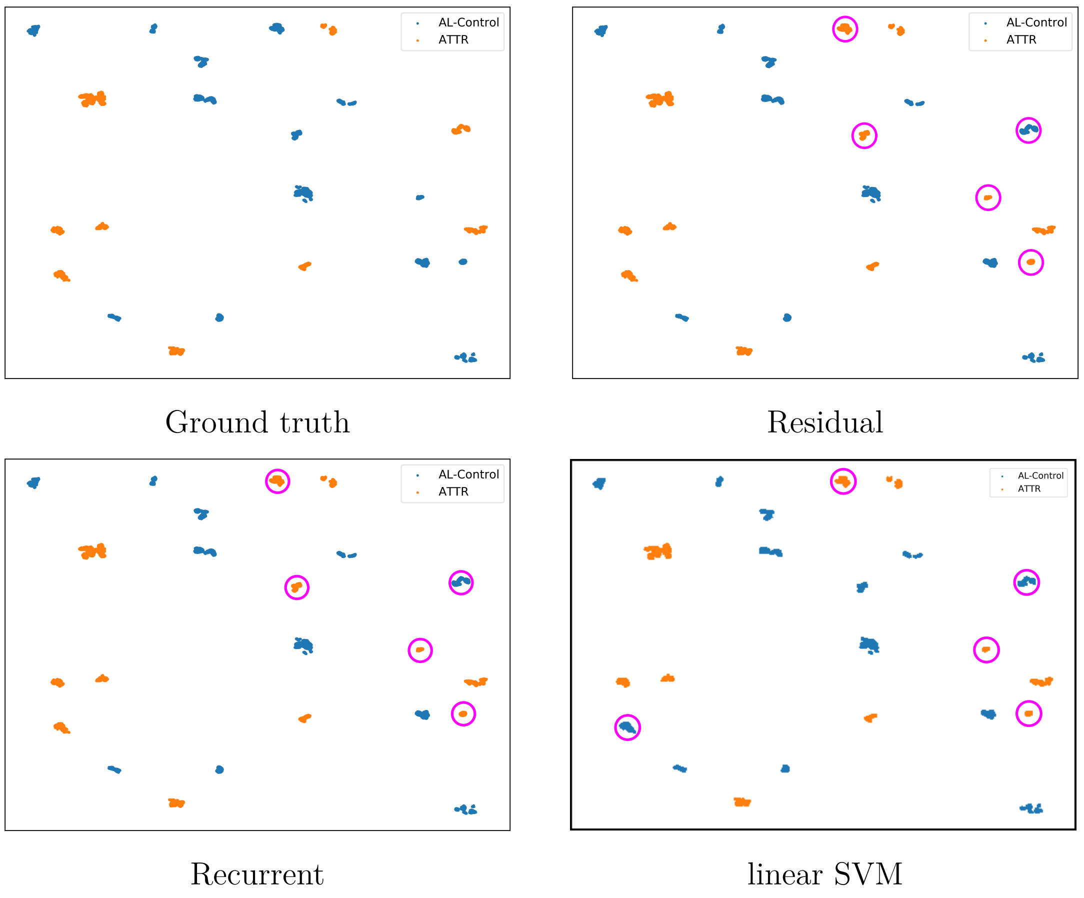


Figure 3.9.: **Top left**: Ground truth of the amyloidosis test split I. **Top right**: Prediction of the Residual model on the patient level. **Bottom left**: Predictions for the Recurrent model. **Bottom right**: Results for the linear SVM. False classified patient samples are marked by a purple circle. All models have an intersection of four patients that are not correctly predicted. Such a result may indicate that the samples in question have characteristics that make prediction difficult. Possible reasons range from an incorrectly transmitted label to low tumor incidence, and the samples are candidates for re-examination by an expert.

Table 3.12.: Results for the best scoring model with randomized labeling in the training set (values for patient in parentheses). The table shows ATTR sensitivity and specificity, the corresponding values for AL-Control are given by the opposite ATTR value. We evaluated the Residual model on a randomized version of all three data splits in Table 3.7. Best results marked in bold numbers.

| Model | Split | Accuracy | Sensitivity | Specificity |
|---|---|---|---|---|
| Residual | random (I) | 0.4046 (0.3478) | **0.8889** | 0.0 |
| Residual | random (II) | 0.5228 (0.5455) | 0.0 | **1.0** |
| Residual | random (III) | **0.5259** (**0.5500**) | 0.0 | **1.0** |

This process is applied only to the training set, keeping the validation and test set untouched and therefore with the correct class labels. This results in a new training set with patient samples having incorrect labeling. The best scoring model from Table 3.10 is evaluated again using the corrupted data. All other parameters are kept fixed and splits are the same as before (Section 3.3.1).

If, as explained in [Des22], information from the test or validation data set would be included in the training, the classification of test data would result in high accuracies. This is because the data set would be fully learned and classification would not be based solely on the incorrect labels in the training data set.

The classification scores for each split drop, displaying a prediction accuracy in the range of 40% to 53% for the given test set (Table 3.12). Again, we consider the two-class problem classifying amyloidosis. The corrupted data causes problems in predicting the patient samples correctly, demonstrating that the presented data splitting prevents the information from leaking into the test set. The results, especially the sensitivity and specificity values, suggest that no meaningful features can be learned from the corrupted data. Instead, a single class is assigned to all samples and an accuracy of about 50% is obtained.

### 3.6.4. Summary and Discussion

We extended the application of neural networks to MALDI-MSI data. For distinguishing the amyloidosis type ATTR from the AL subtype grouped together with amyloidosis plaque-free samples. All network models were evaluated in comparison to the established classifiers as SVM and LDA. Additionally, we tested if feature extraction/ dimensionality reduction benefits prediction precision.

In previous experiments (Section 3.5) network models outperformed other classifiers on MALDI-MSI data. On our amyloidosis data set, the overall accuracy was lower compared to the ovarian cancer data set. Feature extraction generally did not benefit the prediction accuracy, and — previous similar results — the best-performing models were neural networks without feature extraction. The margin to the baseline classifiers widened in the case of amyloidosis plaque classification. This indicates that more complex network models are needed to handle this task which exhibits an even smaller number of samples compared to the ovarian cancer data set, while the feature size remains the same. Interestingly, more complex and deeper model architectures did not increase prediction accuracy. The scarcity of samples in the amyloidosis data set appears to prevent deep models from generalizing well. While there are some features distributed over the whole spectrum, the most important correlations between peaks appear to occur in a more narrow neighborhood.

For the CNN, Recurrent, and Residual models, prediction accuracy dropped after majority voting and computing scores on a patient level. In the case of the Residual model, accuracy dropped by 4% to 75%. For the two-class problem, the decision threshold is set to 50%. Since statistics over a relatively small number of patient samples can easily be affected by a single misclassification, the drop at the patient level can be attributed to these borderline cases. In the future, it could be beneficial to incorporate model confidence into the voting process or the training itself.

The Residual model achieved the best overall performance, followed by a CNN without skip-connections. Compared to the baseline methods, our classifiers achieved an improvement of 4% to 13%.

Winter et al. [Win+17] report remarkable prediction accuracies for the classification of ATTR and AL plaque types. They achieved an accuracy of 93.9% after cross-validation, a potential improvement of 17% over our best model. We believe that multiple factors contribute to this astonishing difference. First, the tissue sections in [Win+17] can be considered to be labeled more broadly. The authors do not use the MALDI-MSI spectra themselves to classify large tissue sections but a statistic over a rather low-dimensional space of six predetermined features. Whereas our data attribute a label to every measurement and, as discussed in Section 3.3.1, this leads to strong noise artifacts and potentially mislabeled measurements containing no plaque at all. Second, we refer to the work of [Des22], which points out the possible pitfalls when testing the distinctiveness of single peaks before classification. The hand-designed filter to extract reliable features for ATTR classification brings the risk of information leaking out of the test set. In general, we consider the process of evaluating the classification performance of possible features before the classification stage questionable. We agree with the findings in [Des22] that this practice is highly prevalent in MALDI-MSI studies and can lead to unreliable studies with impressively high reported accuracies up to 100%. For this reason, we took care to clearly separate training and validation from test data. We remark that [Des22] cite our publication [Kle+19] as an example of good practice in the field of MALDI-MSI.

## 3.7. Pancreas

We further investigate architectural designs for neural networks in MALDI-MSI using a pancreatic cancer data set. It is the largest MALDI-MSI data set yet with hundreds of patient samples and a large feature space with a spectra length of 53 397 $m/z$ positions. In previous sections, we were able to demonstrate that neural networks yield higher accuracy applied to MALDI-MSI data than their established linear counterparts. Therefore, we

Table 3.13.: Data set sizes after random split (percentage of total in parentheses). Samples from individual patients are distributed across the three data sets. Care is taken to ensure that approximately 50% of the patients are used as training data, 20% are used for evaluation during training, and 30% are used in the test data set. The training and test data sets are preferentially populated, so the evaluation data set can be smaller.

| | Training | | Validation | | Testing | |
|---|---|---|---|---|---|---|
| | Patient | Spectra | Patient | Spectra | Patient | Spectra |
| Data set I | 267 (51.1%) | 14348 (50.3%) | 100 (19.2%) | 5578 (19.5%) | 100 (19.2%) | 8627 (30.2%) |
| Data set II | 270 (51.7%) | 14323 (50.2%) | 97 (18.6%) | 5611 (19.7%) | 155 (29.7%) | 8619 (30.1%) |
| Data set III | 254 (48.7%) | 14297 (50.1%) | 102 (19.5%) | 5577 (19.5%) | 166 (31.8%) | 8679 (30.4%) |

focus on the Transformer, as well as the influence of a new preprocessing step to filter non-informative spectra.

### 3.7.1. Data

Pancreatic ductal adenocarcinoma (PDAC) accounts for more than 90% of all pancreatic malignancies and has generally poor prognosis [Kle+16]. We investigate the feasibility of neural networks to accurately classify pancreatic ductal adenocarcinoma based on MALDI-MSI measurements. Tissue samples from 450 patients, diagnosed with exocrine pancreatic cancer, were prepared at the University of British Columbia. The tumor tissue specimens are categorized as ductal adenocarcinoma patients ($n = 261$) and non-ductal ($n = 189$) with the largest subgroup Ampullary carcinoma ($n = 103$). Other pancreatic cancer types are: Acinar cell carcinoma, carcinoma NOS ($n = 1$), chronic pancreatitis (benign, $n = 3$), intraductal papillary-mucinous carcinoma-invasive ($n = 6$), intraductal papillary-mucinous carcinoma-noninvasive ($n = 6$), mucinous cystic neoplasm-noninvasive ($n = 6$), mucinous noncystic carcinoma ($n = 6$), neuroendocrine tumor ($n = 41$), pseudo-papillary tumor, serous cystadenoma ($n = 1$), and signet-ring cell carcinoma ($n = 1$). Peptide signatures extracted from tissue samples yield 434 aligned $m/z$ values in the range of 800–3200 dalton. Table 3.13 shows the data splits for cross-validation.

The study was designed by Steve Kalloger and David F. Schaeffer from the Department of Pathology and Laboratory Medicine, University of British Columbia. The annotation was performed by Axel Wellmann from the Institute of Pathology, Celle. Founding was provided by the BMBF-MSTAR grand.

### 3.7.2. Experiments and Results

In previous sections we developed a process pipeline for MALDI-MSI data, consisting of a filtering step, data set generation for cross-validation and various neural network-based classifiers. This processing pipeline is tested in the context of pancreatic cancer classification, the most challenging task due to its enormous size and molecular similar

Table 3.14.: Class distribution for the metastasis prediction problem. The class Ductal-pN1 accounts for nearly half of all spectra in the data set. The cancer type Ampullary accounts for 40% of spectra with an even split between metastasic and non-metastasic cases. The class shares of the total data set are given in parentheses.

| | Count | Ampullary-pN1 | Ampullary-pN0 | Ductal-pN1 | Ductal-pN0 |
|---|---|---|---|---|---|
| Spectrum | 25812 | 5015 (19.43%) | 5139 (19.91%) | 12165 (47.13%) | 3493 (15.53%) |
| Patient | 630 | 99 (15.71%) | 93 (14.76%) | 335 (53.17%) | 105 (16.67%) |

tissue types. We introduce a more recent model, the Transformer [Vas+17], to MALDI-MSI. The tremendous success of this model for sequential data makes it an interesting candidate for analyzing MALDI-MSI spectra. The Transformer and its various successors [Fuc+20; Dev+18; Bro+20; Dos+21] define current benchmarks in a multitude of tasks such as image captioning, language processing, and image generation. A drawback of the Transformer model is memory consumption (see Chapter 2, Section 2.3). As a result, all experiments are conducted on up-scaled hardware in form of a 2x6-core Intel Xeon Gold 6128 CPU @ 3.40GHz with 24 logical cores and 3x GeForce RTX 2080 Ti GPUs with 11 GB of memory each. We omit the baseline methods due to their inferior performance in the previous experiments and focus on filtering as well as balancing strategies.

**Metastasis Prediction.** The previous success of our models encourages us to test them in a more challenging medical scenario. An influential factor, that heavily impacts the survival rate in cancer treatment, is the development of metastasis. Therefore, metastasis prediction is a highly valuable field of study. Based on the rich metabolic information MALDI-MSI provides, the question arises whether the development of metastasis can be predicted using a combination of spectral imaging and machine learning. Investigation of this question results in a four-class problem, with the classes Ductal-pN0 (no metastasis), Ductal-pN1 (metastasis), Ampullary-pN0, and Ampullary-pN1. One problem that becomes evident immediately is class imbalance. The dominant class Ductal-pN1 has 12 165 samples, nearly half of all samples in the data set. The smallest class Ductal-pN0 accounts for only 3493 samples. Class distributions can be found in Table 3.14. The influence of imbalanced data on the training of neural networks is well documented [HG09].

As a reference point for potential balancing strategies, we train a Residual model, which achieved the best performance classifying amyloidosis plaques (Section 3.6). We employ the two-layer Residual model with kernel sizes of 255 and 127. The number of feature channels is 32 and 16 for the first and second layers. The learning rate is set to $10^{-5}$ and no weight regularization is used.

The results for metastasis prediction are unsatisfactory for this reference model. The mean accuracy over all three splits is 66% on the spectrum and 68% on the patient level.

Table 3.15.: Accuracy for the Residual model for predicting metastasis development from MALDI-MSI data. The first four rows indicate that the model struggles to predict metastasis development. Different balancing strategies are used to counter class imbalance in form of inverse frequency weights (*invBal*) or class member downsampling (*minBal*). Sensitivity and Specificity are provided for class Ductal-pN1 on the patient level. Best results averaged over all splits are marked in bold numbers.

| Model | Split | Accuracy (spectrum) | Accuracy (patient) | Sensitivity | Specificity |
|---|---|---|---|---|---|
| Residual | I | 0.6294 | 0.6503 | 0.8667 | 0.6176 |
| | II | 0.6809 | 0.6966 | 0.8219 | 0.7083 |
| | III | 0.6851 | 0.6993 | 0.9028 | 0.7183 |
| | ∅ | **0.6651** | 0.6821 | **0.8638** | 0.6814 |
| Residual (*invBal*) | I | 0.6116 | 0.6434 | 0.8533 | 0.6471 |
| | II | 0.6813 | 0.6966 | 0.7945 | 0.7222 |
| | III | 0.6825 | 0.7133 | 0.8889 | 0.7606 |
| | ∅ | 0.6585 | **0.6844** | 0.8456 | **0.71** |
| Residual (*minBal*) | I | 0.6123 | 0.6783 | 0.88 | 0.6471 |
| | II | 0.5779 | 0.6138 | 0.8108 | 0.7042 |
| | III | 0.6515 | 0.6853 | 0.8889 | 0.7465 |
| | ∅ | 0.6139 | 0.6591 | 0.8599 | 0.6993 |

In Table 3.15, the sensitivity and specificity values for the dominant Ductal-pN1 class are provided. The unbalanced class distribution results in many samples incorrectly classified as Ductal-pN1, as is evident by the high sensitivity and low specificity.

To counteract the class imbalance, we test two different balancing strategies. First, a weight is assigned to each class to increase the impact of smaller classes during the loss computation. The weight is determined by the inverse frequency of samples in the corresponding class. Second, we force each class to have the same number of samples before constructing the training set. We test maximum and minimum balancing, both balancing strategies are described in greater detail in Section 3.3. Balancing is only applied to the training and validation sets, the class distribution in the test set is unchanged.

Keeping the model design and parameters fixed, we evaluate whether balancing improves the model's performance for metastasis prediction. As can be seen in Table 3.15, balancing does not improve the overall prediction accuracy. Training with *inverse frequency* weights yields a similar level of accuracy, sensitivity, and specificity. For the *minimum* balancing the prediction is even worse, dropping 5% on the spectrum and 2% on the patient level. Overall accuracy is not affected by balancing the class distributions.

Inspection of the confusion matrices in Figure 3.10 reveals two interesting aspects. When comparing the confusion matrix for the model predictions without balancing on the left side and with balancing on the right side, it becomes evident that balancing can

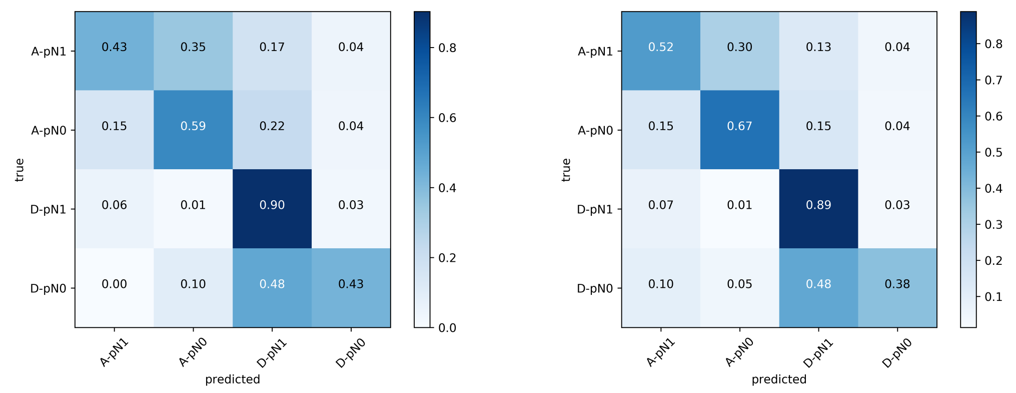


Figure 3.10.: Confusion Matrices for predictions of the Residual model in Table 3.15. **Left**: Residual model without balancing strategies. The dominance of Ductal-pN1 results in high accuracy for that particular class with low accuracies for all other classes. Apart from Ampullary-pN1, all classes are most often wrongfully classified as Ductal-pN1. **Right**: Influence of *inverse frequency* balancing on the prediction accuracy. Balancing is able to reduce miss-classification as Ductal-pN1 for the Ampullary tumor type, but does not change the rate for Ductal-pN0.

reduce the influence of the class imbalance. This corresponds to the shift in sensitivity and specificity for the prediction of Ductal-pN1. But the changes are minor reducing misclassification of Ampullary to Ductal by 4% and 7%. The misclassification within the Ampullary classes benefits the most, gaining 9% in accuracy.

Since balancing does not yield increased accuracy, we investigate adding a special layer to the model. Voting after classification on the individual spectrum often worsens the results. This can be contributed to the low patient count compared to the fast number of spectra in total. Misclassification of one patient sample can lead to a drop in accuracy of multiple percentage points. We employ a **Voting Layer** before loss calculation to shift the model's focus away from the spectra level towards the clinically relevant patient level. The design of the *Voting Layer* is presented in detail in Section 3.3.1.

Again, we use the Residual model to evaluate the influence of the proposed additional layer. To train models with a *Voting Layer* increases instability related to the choice of parameters. The parameter search often produces model configurations, where the loss does not converge during training. These models tend to assign a single class to all samples, which not necessarily has to be the dominant class. The *Voting Layer* often hinders learning, recognizable by low validation accuracy scores during training.

The test results are inferior compared to the original model or balancing strategies. The mean accuracy of approximately 58% is a loss of six percentage points when compared to the original model. Overall it can be said that the current implementation of voting into the training procedure does not increase the model's performance.

Particularly interesting is the fact that misclassification is most prevalent within the tumor type classes (Ductal/Ampullary). We suspect that MALDI-MSI in combination

Table 3.16.: Effects of the Voting Layer. For sensitivity and specificity, only the Ductal-pN1 results are reported. The Voting Layer forces the model to operate on the patient level. Therefore the results for the three split cross-validation are reported on the patient level. Compared to the raw model or balancing strategies, the Voting Layer delivers inferior predictions.

| Model | Split | Accuracy | Sensitivity | Specificity |
|---|---|---|---|---|
| Residual (+ Voting Layer) | I | 0.5594 | 0.7866 | 0.6029 |
| | II | 0.6069 | 0.8513 | 0.5352 |
| | III | 0.5734 | 0.9861 | 0.3943 |
| | ∅ | 0.5799 | 0.8747 | 0.5108 |

with machine learning does not enable reliable metastasis prediction. The reasons can be manifold, including the possibility that proteomic and metabolic patterns are too similar and changes within too minor to be detected. Therefore, we omit metastasis prediction and focus on differentiating between the ductal (PDAC) and non-ductal (non-PDAC) types.

**PDAC vs. non-PDAC.** PDAC and non-PDAC tumors show large differences in patient survival rate and reliable diagnostics to identify PDAC are currently not adequately available. In the following section, we present our work *Classification of Pancreatic Ductal Adenocarcinoma Using MALDI-MSI combined with Neural Networks* [Kan+23], published in *Cancers* Special Issue **Advances in Mass Spectrometry Imaging-Based Cancer Research** 2023.

Current classification strategies for PDAC use a univariate approach by identifying single $m/z$ peak locations, followed by the evaluation of these $m/z$ peaks in their ability to discriminate between pancreatic ductal adenocarcinoma and ampullary carcinoma. This is achieved using *Receiver Operator Characteristics* (ROC) analysis to a total of 435 $m/z$ peak locations from both ductal and ampullary carcinoma. A total of 131 $m/z$ peak locations are identified, expressing discriminative intensity distributions between PDAC and ampullary carcinoma (AC). The discriminative power is measured by the area under the curve (AUC), and a peak is considered if its AUC value is either above 0.7 or below 0.3.

As a result, three proteins are identified, displaying significantly higher intensity distributions in PDAC tumor regions compared to ampullary carcinoma (Figure 3.11). These proteins are promising candidates for potential biomarkers to distinguish PDAC and non-PDAC samples. The seven $m/z$ locations corresponding to the PELC, AHNAK, and COLA3 proteins are listed in Table 3.17. These proteins have known biological functions in PDAC development. PLEC drives proliferation, migration, and invasion [Shi+13]. AHNAK also influences proliferation as well as migration [Zha+19a] and COL6A3 is a potential prognostic factor for PDAC [Svo+20; Kan+14]. Also, these three proteins are potentially interesting biomarkers, the AUC value of approximately 0.7 does not

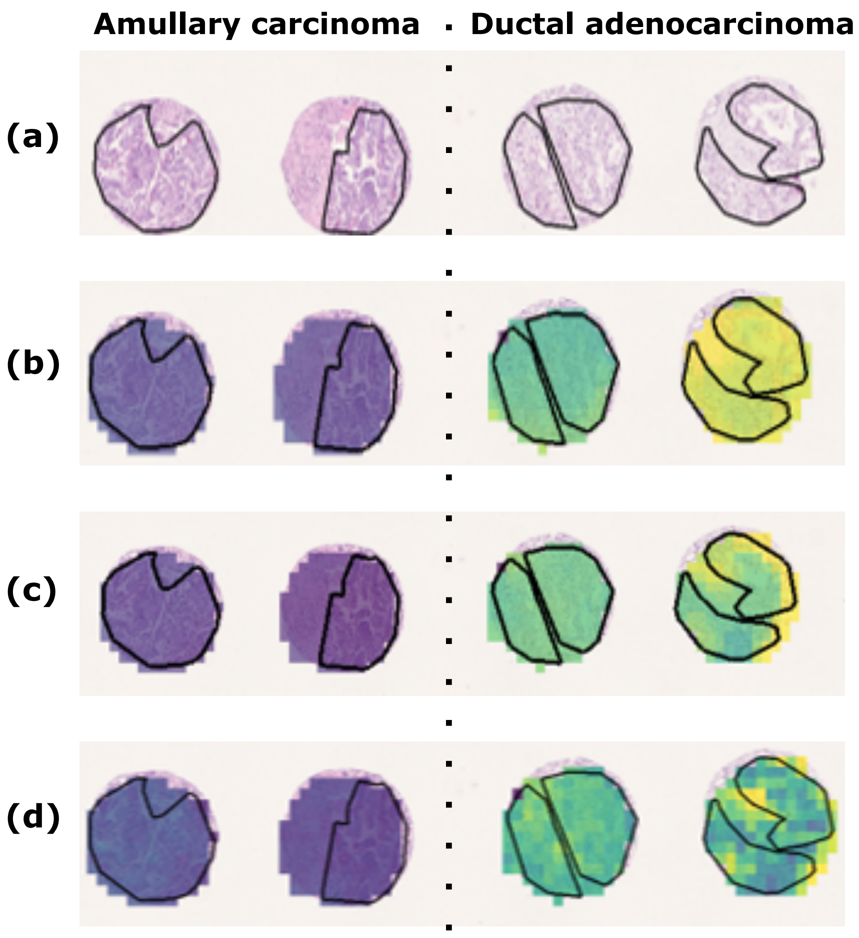


Figure 3.11.: Discriminative protein markers for pancreatic ductal adenocarcinoma and ampullary carcinoma tissue sections. For orientation, results of hematoxylin and eosin (H&E) staining are provided in the first row (**a**). Three marker proteins were identified at the positions (**b**) $m/z$ 1459 assigned to COL6A3, (**c**) $m/z$ 1479 assigned to Plectin and (**d**) $m/z$ 1267 assigned to AHNAK are shown. All three proteins display increased intensity distributions in ductal adenocarcinoma. Black lines delineate the tumor border area. Purple indicates low intensity and yellow indicates high intensity at the corresponding m/z position.

Table 3.17.: Differential intensity distributions of peptides (MALDI-MSI) and their corresponding proteins in tissue sections from pancreatic ductal adenocarcinoma and ampullary carcinoma tissue (tumor regions).

| $m/z$ value | AUC (AC/PDAC) | Protein |
|---|---|---|
| 1459.7 | 0.721 | Collagen alpha-3(VI) (COL6A3) |
| 1586.8 | 0.700 | |
| 1267.7 | 0.715 | AHNAK |
| 1655.8 | 0.719 | |
| 1461.7 | 0.710 | Plectin (PLEC) |
| 1479.8 | 0.714 | |
| 2115.1 | 0.701 | |

support PDAC assessment in a clinical setting. Furthermore, several studies demonstrate that multivariate statistical treatment of MALDI-MSI data achieves superior results compared to the baseline univariate marker identification [McC+05; DBS10; Jon+11; Ves+14]. Multivariate statistics utilize the complete spectral information, thus obtaining differences that are undetectable using univariate methods. Machine learning algorithms have the potential to explore statistical correlations, find similarities within spectral subgroups, and use these findings to classify the data.

In order to do so, we apply neuronal network models to the pancreatic cancer data set. Our experiments for the metastasis prediction demonstrated that class imbalance is a major concern while working with the data. Therefore, we aim to reduce the impact by performing the classification of PDAC ($n = 260$) against all non-PDAC ($n = 189$, including 103 ampullary carcinoma patients). We employ several models as classifiers: a two-layer Residual model and an encoder-only variant of the Transformer architecture [Vas+17], see Figure 3.12 for the modified version.

The Residual model is used as a baseline, due to its performance in classifying amyloidosis plaque types. For the Transformer model, the size of the attention matrices is $n \times n$, where $n$ denotes the sequence length. A spectrum consists of several thousand data points, rendering the application of a Transformer to the full-scale spectrum unfeasible. Consequently, we reduce the sequence length using a preceding pooling layer with a kernel size of 4. We choose the size based on the scale of a single peak, consisting of 3 individual data points. After a parameter search over the learning rate, batch size, channel dimensions, and the number of heads, we arrive at two model configurations that perform best on the test set, which we refer to as Transformer-1/2 (Table 3.18). Again, we initialize the model weights by randomly selecting values from a truncated normal distribution (Xavier). As activations, the Rectified-Linear-Units (ReLU) function is used as well as the Adam optimizer.

During the random search over parameter configurations, we observe a high sensitivity to the choice of the learning rate for both the Residual and Transformer model. For this reason, the best-performing models use similar learning rates of $10^{-4}$ to $10^{-5}$. Rates

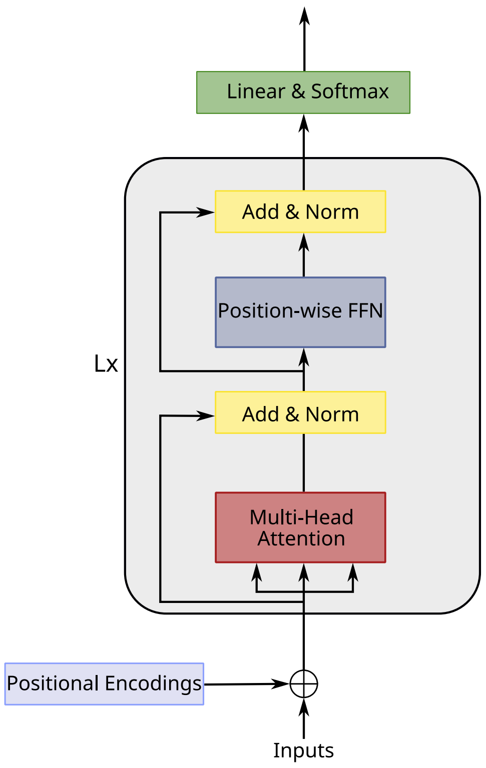


Figure 3.12.: Transformer architecture in the encoder-only variant utilized in this thesis. The original design in [Vas+17] is largely kept with minor changes to the input embedding: The MALDI-MSI spectrum can be seen as a histogram correlating mass counts to detection time. These spectra are by nature numeric data and do not need to be embedded, as is the case with text. As a result, the embedding layer can be omitted and the inputs fed raw to the Transformer-Encoder. We enable classification by adding the softmax layer from the original Transformer decoder path on top of a multi-layer encoder structure. The *Lx* indicates that the grey encoder block can be repeated multiple times and all Transformer models in this work are composed of the here displayed encoder building block.

Table 3.18.: Parameter configurations for the Residual and Transformer models. The number of heads is exclusive to the Transformer architecture.

| Model | Kernel Size | Channel | Learning Rate | Batch Size | Heads | Pooling Size |
|---|---|---|---|---|---|---|
| Residual | (200, 100) | (16, 8) | $10^{-5}$ | 100 | - | - |
| Transformer 1 | (256, 256) | (16, 8) | $10^{-4}$ | 100 | 2 | 4 |
| Transformer 2 | (512, 512) | (16, 8) | $10^{-4}$ | 500 | 2 | 8 |

Table 3.19.: Results for neural network models. Given are the model classification accuracies for spectrum and patient predictions. Best accuracy in bold numbers.

| Model | Split | Accuracy (spectrum) | Accuracy (patient) |
|---|---|---|---|
| Residual | I | 0.86 | 0.86 |
| | II | **0.76** | **0.77** |
| | III | 0.77 | 0.76 |
| | ∅ | 0.80 | 0.80 |
| Transformer1 | I | 0.85 | 0.86 |
| | II | **0.78** | **0.77** |
| | III | 0.77 | 0.77 |
| | ∅ | 0.80 | 0.80 |
| Transformer2 | I | 0.83 | **0.84** |
| | II | **0.77** | 0.76 |
| | III | 0.78 | 0.79 |
| | ∅ | 0.80 | 0.80 |

outside of this range cause the loss either to decrease (higher rates) or to not converge at all (lower rates). The other parameters, i.e., batch size and channel dimension, do not show the same effect. Increased network depth does not improve accuracy but ultimately harms performance.

Application of the noise filter (Section 3.3.1) results in an increase of 3% in accuracy for all tested models. The filter excludes 2183 grid points, accounting for approximately 13% of spectra. The Transformer model achieves an accuracy of 85% on the spectral level and gains an additional percent if evaluated on the patient level (see Table 3.19). But the best performance is again achieved by the Residual model with an accuracy of up to 86% on both the spectral and patient level. Averaged over the 3-fold cross-validation, the Residual model achieves an accuracy of 80% on both levels. The Transformer attention mechanism seems to not improve on the prediction compared to the Residual model's skip connections.

### 3.7.3. Summary and Discussion

Univariate methods and single protein marker classification do not produce sufficient accuracies for PDAC classification. This can be attributed to tissue heterogeneity and the large data size complicating reliable identification of relevant molecular patterns. We demonstrated that neural networks offer an alternative to univariate approaches. Our models were able to differentiate between PDAC and non-PDAC tissue with an accuracy of up to 86% and sensitivity of 82% and allow for an accurate and quick prediction of large data sets with limited preprocessing.

We evaluated the performance of two model architectures: a Residual model, based on the original ResNet architecture [He+16], and the popular Transformer model [Vas+17] in an encoder-only variant. The Residual model achieved the best performance on a single split but overall both architectures were able to predict PDAC with an accuracy of 80%.

Additionally, we proposed a preprocessing step to tackle the problem of noise due to structurally non-informative spectral data points. We addressed this caveat, utilizing only properties of the spectrum itself without the need for user interaction. Our filter reduces noise by excluding spectra before the classification step. In the case of the pancreatic cancer data set, we were able to identify 2183 spectra with low informativeness, resulting in an increased accuracy of 3% across all models.

## 3.8. Conclusion

We evaluated the **application of neural network-based classification** combined with the **MALDI-MSI** technique. Neural networks proved to be a good alternative to the currently employed classification algorithms, often improving on the results these methods are able to achieve. These results encourage a more frequent usage of neural networks in the context of MALDI-MSI.

In Section 3.5 we employed neural networks to classify four subtypes of ovarian cancer. These subtypes share many morphological structures and classification thereof is regarded as a complex task. In contrast to predeceasing studies, we advocate utilizing the full spectral range instead of commonly-used feature extraction. Neural networks express great capabilities for automatic feature extraction, due to their layered structure. The hierarchical design allows the modeling of more complex features.

As mentioned in Section 3.2, so far, few works applied neural networks to raw spectral input. The IsotopeNet [Beh+17] is characterized by a thorough analysis of the morphological/proteomic structures in the given data set. As a result, the network model achieves high prediction accuracies. But, as we demonstrated in Section 3.5, is not seamlessly adaptable to any MALDI-MSI data set. The basis of the IsotopeNet is a Residual network with skip connections, which we use as inspiration for later model designs in Sections 3.6, 3.7. In accordance with the findings in this thesis, the work in [Mit+21] also successfully employed neural networks in a MALDI-MSI setting. The authors state that few-layer models are sufficient to outperform decision tree algorithms, similar to our observations concerning the relationship between model depth and prediction performance.

For the ovarian data set, our models achieved the best performances in terms of prediction accuracy, proving superior to established methods, such as LDA and SVM.

Commonly machine learning algorithms are combined with feature extraction when employed in MALDI-MSI, which can either be automatic (e.g. PCA, NMF) or hand-designed (peak picking). We demonstrated in Section 3.6 that feature extraction harms the network's performance in many cases and the best results are achieved when omitting feature extraction.

At this point it is necessary to mention a common mistake when working with hand-designed features: In many studies utilizing a combination of machine learning and MALDI-MSI, data sets are not sufficiently separated. This leads to unreasonably high prediction rates while working with manual features, as demonstrated in [Des22]. In our experiments, the implementation of data splitting based on patients prevents information from leaking into training. We also advocate against testing the separation capabilities of features before the classification step, a common practice while working with MALDI-MSI data.

Although reliable metastasis prediction currently appears to be out of reach (Section 3.7), we successfully applied our models to the classification of pancreatic ductal adenocarcinoma, a prevalent and difficult-to-diagnose disease. We found that MALDI-MSI data contains an additional noise source, apart from the ones described in the literature [Dei+11; Nor+07]. We designed a filter to remove spectra with low informativeness as a preprocessing step, which resulted in an increased accuracy of 3% for the PDAC classification. Moreover, we tested the recent Transformer architecture in the field of MALDI-MSI, demonstrating performance on the level of our best-performing model.

# Part II.

# Meta-Learning for Image Registration

# 4. Image Registration Fundamentals

In this chapter, we focus on image registration as well as the registration framework used in this thesis. First, we give a general introduction to the concept of image registration in Section 4.1, which serves as a basis to better understand the newly developed network-based methods presented in Chapter 5. Discretization is discussed in Section 4.2 and relevant optimization methods are presented in Section 4.3 along with the line search and multi-level scheme. Optimization is central to the methods developed in this thesis, which serve as a bridge between classical and learning-based approaches, as the classical formulation provides the basis on which we build our methods. Section 4.4 provides an overview of the benchmark toolboxes, which will be used in Chapter 5 to compare our methods to baseline variational methods.

## 4.1. Image Registration

In the following, we expand on the brief introduction to image registration in Chapter 1 and the implementation details. We focus on the alignment of two gray-valued images, denoted as a fixed reference image $\mathcal{R} : \mathbb{R}^2 \to \mathbb{R}$ and a template image $\mathcal{T} : \mathbb{R}^2 \to \mathbb{R}$. A deformation field $\varphi : \mathbb{R}^2 \to \mathbb{R}^2$ maps points from the reference to the template image domain. The goal is to find $\varphi$ so that the deformed template image $\mathcal{T} \circ \varphi$ is similar to the reference image regarding a given similarity metric. We can differentiate between two forms of transformation, globally operating affine and local nonlinear (deformable) deformations. Finding a suitable transformation function is classically formulated as an optimization problem,

$$\min_{\varphi} f(\varphi), \quad f(\varphi) := \mathcal{D}(\mathcal{R}, \mathcal{T} \circ \varphi) + \gamma \mathcal{S}(\varphi), \tag{4.1}$$

with a similarity metric $\mathcal{D}$, a regularizer $\mathcal{S}$, and a tuneable weight $\gamma > 0$. The objective function $f$ often displays multiple local minima. Therefore, developing robust optimization algorithms is a field of intense research in image registration. Multiple methods aim to solve the optimization problem in (4.1). We categorize these into classical variational methods [Ami94; Mod03; Bre+19; DR04; DGM98] and neural network based approaches [Bal+19; She+19; MC21b; BHH21].

## 4.2. Discretization

We follow the *discretize-then-optimize* approach used in *FAIR* [Mod09]. The objective function $f$ and transformation $\varphi$ are discretized on regular grids of equidistant points on the image domain

$$\Omega_{\mathcal{R}} := (\omega_1, \omega_2) \times ... \times (\omega_{2d-1}, \omega_{2d}) \subset \mathbb{R}^d. \tag{4.2}$$

The location of grid points in the discretized domain can differ depending on the grid. Prominent examples of grid types in the literature are cell-centered, nodal, and staggered grids. Depending on the discretization and desired properties, for example, during image interpolation, a cell-centered grid can be employed [HM06]. In this thesis, we discretize images and transformations on cell-centered grids, which are well-suited for our use case. Since they are easy to use and allow us to take advantage of the automatic differentiation tools of the *PyTorch* framework. The number of cells in each dimension is given by $\bar{m} := (m_1, m_2, ..., m_d)$ and the size of a cell is determined by the spacing $h := (h_1, ..., h_d)$. The cell center coordinates are defined as

$$x_j := \left(\omega_1 + h_1\left(i_1 - \frac{1}{2}\right), ..., \omega_{2d-1} + h_d\left(i_d - \frac{1}{2}\right)\right), \tag{4.3}$$

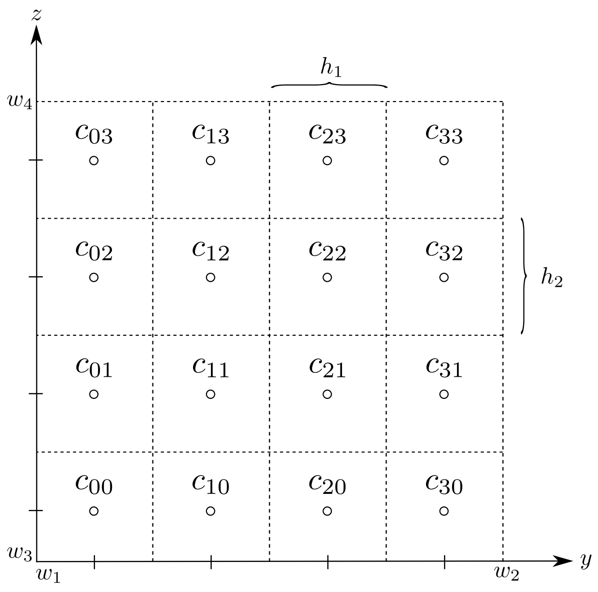
with multiindex $i = (i_1, ..., i_d)$, $i_l = 1, ..., m_l$ for $l = 1, ..., d$. We arrange all coordinate components in a lexicographical fashion and store them in a single vector $x =\in \mathbb{R}^{dm}$. See Figure 4.1 for a visualization of a cell-centered grid. We can discretize an image $\mathcal{I}$ in a similar fashion and define a function

$$I(x) := (\mathcal{I}(x_k))_{k=1,...,m} \in \mathbb{R}^m \tag{4.4}$$

mapping the coordinates to a vector of image intensities. Here $m = \prod_k m_k$ denotes the total number of cells. The given grid notations are used to define our loss function in Chapter 5. For a more detailed discussion of grids see [Pap08] and [Mod09].

**Interpolation.** In practice, images are discrete data on a regular grid of pixels or voxels. Computing the deformed image $\mathcal{T}(\varphi)$ requires evaluating the image at any point of the continuous domain $\Omega_{\mathcal{T}}$ which requires *image interpolation.* Various methods can be used, such as nearest-neighbor-interpolation [TBU00] or spline-based models [LGS99]. We employ standard bilinear interpolation following [Mod09]. Linear interpolation is commonly used in the context of medical image registration [Sha+10]. Note that the interpolation is ideally differentiable when using gradient-based update schemes, such as backpropagation, in neural networks. Although linearly interpolated images are not ensured to be continuously differentiable we opted in favor of fast and efficient computation in this case.

**Distance Measure.** We established that the energy function in (4.1) consists of two parts, the distance metric and a regularizer. The *distance metric* or *distance measure* defines image similarity. In this thesis, smaller distance values correspond to a better alignment of images. We adopt the categorization established in [Kön18] and distinguish between *mono-modal (unimodal)* and *multi-modal* distance measures. For the sake of completeness, we discuss both variants but solely employ *mono-modal* distance measures in Chapter 5. If the images both originate from the same acquisition device *(mono-modal)*,



$z$
$h_1$
$w_4$
$c_{03}$ $c_{13}$ $c_{23}$ $c_{33}$
$c_{02}$ $c_{12}$ $c_{22}$ $c_{32}$
$h_2$
$c_{01}$ $c_{11}$ $c_{21}$ $c_{31}$
$c_{00}$ $c_{10}$ $c_{20}$ $c_{30}$
$w_3$
$w_1$
$w_2$
$y$

Figure 4.1.: Cell-centered grid on the domain $\Omega_{\mathcal{R}}$ for a two-dimensional image ($d = 2$). The grid points $c_{ij} = (y_i + \frac{h_1}{2}, z_j + \frac{h_2}{2})$ are located at the center of the cells. The user can choose the cell size ($h_1$,$h_2$), for full-scale images the number of cells is usually the same as the number of pixels.

this allows us to directly compare the two images. The *sum of squared differences* (SSD) is a frequently used distance metric. It is easy to implement, fully differentiable, and therefore a well-suited candidate for gradient-based methods in the form of (4.7) or neural networks with their backpropagation update [RHW86]. More advanced variants of the SSD metric are described in [Pli+08; Hil+01; Du+16].

The discretized SSD metric is

$$\mathcal{D}^{\mathrm{SSD}}(\varphi) := \tilde{h} \sum_{i=1}^{m} ((T \circ \varphi)_i(x)) - R_i(x))^2, \tag{4.5}$$

where $\tilde{h} = \prod_d h_d$. The terms $(T \circ \varphi)(x)$ as well as $R(x)$ are the discretized deformed template and reference image as described in (4.4). A discussion of the continuous SSD measure can be found in [Mod03].

In medical image analysis, we often compare images from different acquisition devices. As a result, similar structures in the images can vastly differ in image intensity and prohibit the deployment of methods measuring image similarity by point-wise comparison. Alternatively, *multi-modal* distance metrics can be used. A multitude of methods exist in form of maximization of correlation ratio [SDP13], measurement of image gradient alignment [HM05; HM06], or modeling similarity by a shared distribution of random variables [Mae+97]. For a more complete overview of common distance metrics, the reader is referred to [KBD17; Sha+10].

**Curvature Regularizer.** As mentioned in Section 4.1 the second part of our energy function is a *regularizer*. It is a crucial component of the variational model, especially for non-parametric deformation. Registration problems usually are ill-posed [Had02] without a unique and stable solution [Mod03]. Regularization favors specific properties of the deformation and is regularly based on a physical model of motion [FM02; BMR13]. It does not depend on the image data in (4.1), but only on the displacement itself. The *curvature* regularizer [FM03] is popular for real-world problems [Bre+19; Her+19; Pol+16] and is used throughout all experiments in Chapter 5.

Fisher and Modersitzki introduced the *curvature* regularizer in 2003 [FM03]. It is based on the second-order derivatives of the displacement. Given a displacement $u(x) := \varphi(x) - x$, the discrete formulation of the curvature regularizer is

$$\mathcal{S}^{\text{curv}}(\varphi) = \tilde{h} \sum_{i=1}^{m} \sum_{j=1}^{d} (\Delta u)^2_{i+(j-1)m}. \tag{4.6}$$

Here $\nabla^2 u$ is a finite-differences approximation of the Laplacian. The non-trivial kernel of the *curvature* regularizer contains the affine transformations reducing the need for pre-alignment and favoring smooth deformations.

## 4.3. Numerical Optimization

Classical registration algorithms for obtaining a suitable deformation are commonly based on numerical optimization methods [SDP13; Mod03]. In this thesis, we focus on these methods, which aim to find an approximate minimizer of (4.1). These gradient-based solvers produce iterative update steps in the form of

$$x_{k+1} = x_k - \alpha_k B_k \nabla f(x_k) \tag{4.7}$$

with a step length $\alpha_k$ and a preconditioning matrix $B_k$. When choosing $B_k$ to approximate $(\nabla^2 f(x_k))^{-1}$, this results in a "Quasi-Newton Method" [NW06, Chap.6], see also [DS96, Chap.9]. Note that the calculation of the conditioning matrix $B_k$ can be very time and memory-consuming. For example, the Newton method requires us to compute the exact Hessian of the objective function in every iteration and solve a system of linear equations based on it. An alternative is the use of an approximated Hessian, for instance in the popular Broyden-Fletcher-Goldfarb-Shanno (BFGS) method. We apply the BFGS method and especially its adapted limited-memory version for most of the experiments in Chapter 5 and therefore discuss it in more detail here.

**BFGS Method.** Instead of computing the next update using $B_k$, which would require solving a linear system of equations at each step, an update can be derived for

$$H_k = B_k^{-1}, \tag{4.8}$$

the inverse approximate Hessian containing the second-order-derivatives for $f$ in (4.1) at the current iterate $x_k$. This formulation allows the calculation of the search direction by means of a matrix-vector multiplication. As mentioned above, the BFGS method is a Quasi-Newton method only requiring the gradient of the objective function at each iterate [NW06, Chap.6]. For large problems, computing $H_k$ is not feasible due to the high computational costs. The *limited-memory BFGS* (L-BFGS) does not store the full dense matrix but instead uses an implicit approximation represented by a few vectors. These vectors contain curvature information only from the most recent iterations. Following the formulation in [NW06, Chap.6] and $\nabla f_k := \nabla f(x_k)$, the update for the limited-memory BFGS takes the form

$$x_{k+1} = x_k - \alpha_k H_k \nabla f_k. \tag{4.9}$$

and the inverse Hessian $H_{k+1}$ can be directly updated by

$$H_{k+1} = \left( I - \frac{s_k g_k^T}{g_k^T s_k} \right) H_k \left( I - \frac{g_k s_k^T}{g_k^T s_k} \right) + \rho_k s_k s_k^\top \tag{4.10}$$

where

$$\rho_k = \frac{1}{g^\top s_k} \tag{4.11}$$

with $s_k = x_{k+1} - x_k$ and $g_k = \nabla f_{k+1} - \nabla f_k$ the differences in the iterates and gradients thereof. Here $I$ is the identity matrix. We can limit the number of vectors $s_k$ and $g_k$ to $k \in \{K-1, ..., K-n\}$, where $K$ is the current step and $n \in \mathbb{N}$ the number of past steps we want to include, in this thesis we use $n$=5. In the first iteration, an initial $H_0$ is required, we follow the guidelines in [NW06, Chap.6] and set it to

$$H_0 = \frac{1}{\nabla f(0)} I. \tag{4.12}$$

For more details, we refer to [DS96; NW06].

**Line Search.** Finding a suitable step length $\alpha > 0$ in (4.7) is crucial for the L-BFGS method. Various *line search* algorithms exist to find $\alpha$ so that the objective function value is sufficiently reduced. Exact line search algorithms test multiple step lengths to determine an exact minimum, which can be very time-consuming. In contrast, inexact line search algorithms only require $\alpha$ to fulfill some weaker conditions. One popular example is the curvature condition [NW06, Chap.3], which ensures the slope of the objective after the update is reduced by at least some given factor. In this thesis we employ the *Armijo condition* [NW06, Chap.3] as proposed in [Mod09], which ensures a sufficient decrease in the objective function. We also follow the implementation in [Mod09], starting with $\alpha = 1$, reducing the step length by half until the condition is met or a maximum of ten iterations is reached. This serves as a safeguard and stops the optimization, preventing an update along a non-descent direction causing the optimization to fail. As a result, the registration is not accurate and usually demands user interaction for the given image pair. This drawback of line search will be discussed in Chapter 5.

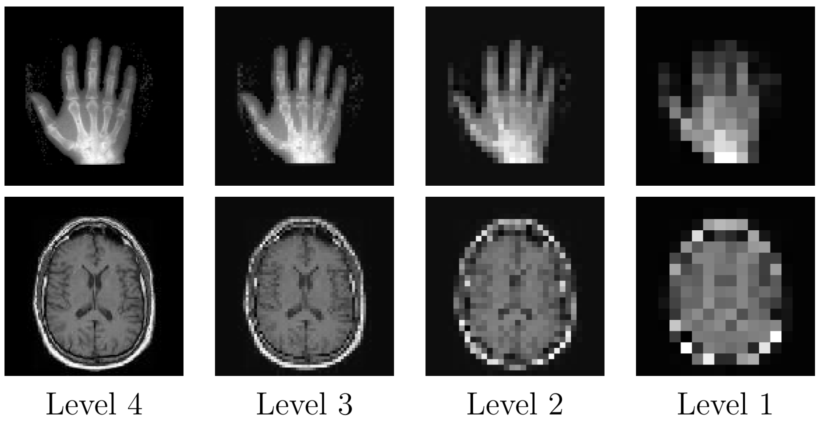


Figure 4.2.: Multi-level representations of two images. **Top**: X-ray image of a hand with a starting resolution of $128 \times 128$ pixels on the left (level 4), in each level the image size is halved. The last image on the right has a resolution of $16 \times 16$ pixels (level 1). **Bottom**: The ML representation for a brain MRI image, also starts from $128 \times 128$ pixels with a final resolution of $16 \times 16$. The low-resolution representations (right) produce fewer local minima in the distance function, due to the reduced detail present in the images. This enables better compensation of large deformations on coarse image representations.

**Multi-Level Scheme.** The objective in (4.1) often is non-convex and rarely has a global minimum but commonly contains multiple local minima. Iterative gradient-based methods operate in a local fashion and tend to get stuck in local minima. The solution may not correspond to the desired deformation. A solution to this is a coarse-to-fine *multi-level scheme*, solving the registration problem on different resolutions, starting with a coarse image representation (Figure 4.2). The objective function generally has fewer local minima on a low-resolution image representation.

We compute image representations of different resolutions by successively sub-sampling the images to a size of $m_{\text{coarse}} = \lfloor \frac{m}{2} \rfloor$ for both reference and template images. Sub-sampling is achieved using linear interpolation described in Section 4.2. Repeating this process results in a sequence of differentially coarse resolved image representations. The corresponding deformation grids are sub-sampled in the same fashion to maintain the original ratio, between grid and image. Additionally, a low-pass filter can be applied before sub-sampling [Jäh05; Mod09] to further smooth images and consequently reduce the number of local minima in the objective function, as the smoothing of images eliminates detailed structures.

The registration problem is solved iteratively, starting with the lowest resolutions for the image and deformation grid. As an initial guess for the first iterate, either the identity transformation or the result of another registration can be used. An upsampled version of the current level's result is then used as a starting point for the next finer level. In order to obtain the versions of the images with different resolutions, we use the bicubic variant of the interpolation methods of PyTorch.

## 4.4. Benchmark Toolboxes

For comparison purposes, we will review the *FAIR* [Mod09] and *elastix* [Kle+10] toolboxes. Both toolboxes provide a modular structure to enable the user to design and apply a registration pipeline for any given problem. We will briefly discuss both variants and their suitability as benchmark methods.

**FAIR.** *FAIR* stands for *Flexible Algorithms for Image Registration.* It provides state-of-the-art numerical methods as well as registration techniques implemented using the MATLAB [MAT18] software environment. *FAIR* is originally designed for educational purposes with a special emphasis on the theoretical background. In *FAIR*, the registration problem is phrased in a variational setting. An in-depth description of all registration pipeline modules, such as transformation models, distance measures, regularizers, and numerical tools, can be found in [Mod09].

Although Python-based our own implementations follow the formulations in *FAIR*, consequently the algorithms depicted in Sections 4.2 to 4.3 are also implemented in *FAIR* using the same *discretize-then-optimize* approach.

Given the modular structure of *FAIR*, the user has many options to customize the registration setting. In this thesis, we aim to compare a classical with a learned optimization. Therefore, we want to keep the setting as similar as possible, adjusting the following modules:

- The transfomation is set to either `affine2D` or `nonpara2D` accordingly to the setting in our network model.
- The distance $\mathcal{D}$ is set to `SSD`, which is the distance function used in our experiments in Chapter 5.
- We enable the multi-level scheme with 4 levels of resolution. *FAIR* provides a kernel option to smooth the multi-scale images, which we set to `gaussian`.
- We chose the regularizer $\mathcal{S}$ to be `mbCurvature`, due to the reasons provided in Section 4.2. The *mb* stands for matrix based, clarifying the method of implementation.
- For default optimizer, we choose `limited BFGS`.

**Elastix.** *Elastix* [Kle+10] is a toolbox for image registration based on the well-known Insight Segmentation and Registration Toolkit (ITK). It is designed with a medical use case in mind and provides a collection of algorithms similar to the design of *FAIR*. The *SimpleElastix* extension makes it available in a variety of programming languages such as Python, which enables effortless integration in our existing framework.

*Elastix* is characterized by modular design similar to *FAIR*. We use the grid search technique to find suitable parameter configurations for our tasks in Chapter 5. An overview of the most important components of *elastix* is depicted in Figure 4.3.

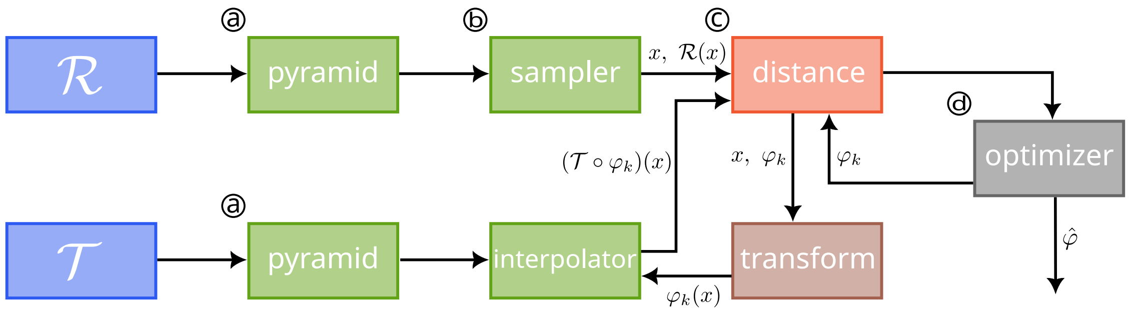


Figure 4.3.: Illustration of *elastix* components [Kle+10]. The parameters the (a) *pyramid*, (b) *sampler*, (c) *distance*, and (d) *optimizer* modules are adjusted using grid search. Here $\mathcal{T}$ and $\mathcal{R}$ are the template and reference images, $x$ denotes the grid vector, and $\varphi_k$ is the transformation at step $k$. The final transformation is denoted by $\hat{\varphi}$

.

Given the task requirements, we select a subset of options, that we consider useful to test such as Gradient Descent and L-BFGS. We compare our network models to the configuration with the smallest error on the test set. Here the error is the *Mean Square Error* between the predicted and the ground truth deformation, evaluated on the grid. The *transform* module defines whether an affine or non-parametric transformation is considered. The transformation module is set accordingly to the given task in Chapter 5. We consider the following modules, parameter options are given in parentheses:

(a) We discussed the multi-level image representations in Section 4.3. In *elastix*, these are set by the *image pyramid* module. We apply a smoothing kernel to reduce the number of minima in the distance function (`FixedSmoothingImagePyramid`, `MovingSmoothingImagePyramid`).

(b) The *sampler* defines the positions used to evaluate the current transformation using the distance measure. More points provide a more accurate evaluation at the cost of longer calculation times (`Grid`, `RandomCoordinate`).

(c) We partly divert from our settings in *FAIR* and test multiple distance *metrics* (`AdvancedMeanSquares`, `AdvancedNormalizedCorrelation`).

(d) Similar to the metric module, we allow for an expanded selection of choices regarding the *optimizer*. Apart from two L-BFGS variants, we also included the more recent AdaGrad and an adaptive SGD (`AdaGrad`, `AdaptiveStochasticGradientDescent`, `QuasiNewtonLBFGS`, `AdaptiveStochasticLBFGS`).

We compare both benchmark methods to our network-based approach in Chapter 5.

For completeness, we also mention the *Advanced Normalization Tools* (ANTs) [ATS+09] framework, which has gained increased popularity in recent years. It provides a variety of methods with a focus on deformable image registration and originally was developed with neuroimaging data in mind.

## 4.5. Neural Networks for Image Registration

Neural networks, especially CNNs, are also increasingly used in image registration. Today, they account for many newly developed methods [Bal+19; Roh+17; Cao+17; Vos+17] and nearly reach the same level of accuracy as classical methods [Xu+21]. This has two main reasons: First, the success of networks in other areas of computer vision indicates their potential in image registration. Secondly, they allow for fast registration of new images in a single forward pass. Classical methods with their iterative approach require considerably more time to process unseen image pairs. We discuss a few important representatives of convolutional neural networks, as well as the more recent iterative and recursive network variants.

First, we consider networks that generate a transformation in a single forward pass without inspiration from classical methods, such as iterative updates or a variational setup.

**Spatial Transformer.** In [JSZ+15], the *Spatial Transformer* module is introduced (Figure 4.4). It enables CNNs to learn invariance to translation, scale, and rotation. The network module spatially transforms an image or feature map by producing transformations for each input sample. Incorporated into a CNN, the module allows the network to learn to transform its input and minimize the network's loss function. The most important property is the differentiability of the sample grid, which allows the loss gradient to flow back to the transformation parameters. The module is split into three parts: a localization network, a grid generator, and a sampler. The *localization network* converts the input into parameters of a spatial transformation. This step produces a transformation conditioned on the input. Next, a sampling grid is created using the transformation parameters. This is done by the *grid generator*. Finally, the feature map and sampling grid are passed to the *sampler* to create the transformed output. Although the *Spatial Transformer* is originally designed to boost the performance of existing models in tasks such as classification or co-localization.

**VoxelMorph.** The *VoxelMorph* [Bal+19] model is a fully convolutional network that uses the *Spatial Transformer* to establish a purely learning-based registration method. Unlike classical methods, VoxelMorph formulates the registration as a function that maps the input image pair directly to a transformation. For this purpose, the network uses the template $\mathcal{T}$ and reference image $\mathcal{R}$ as inputs to calculate the transformation $\varphi$. The mapping function is parameterized by the network parameters $\theta$. This step is performed by a *UNet* [RFB15] architecture. The pyramid structure of the *UNet* generates differently resolved representations of the input images and fulfills a similar function

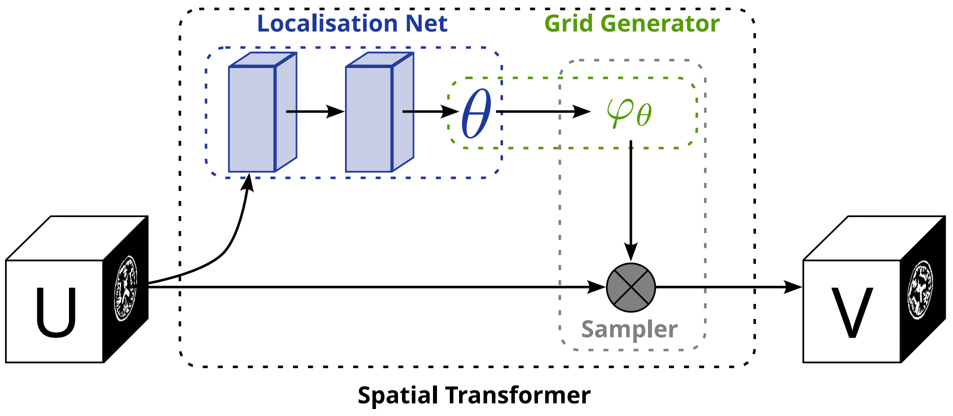


Figure 4.4.: Visualization of the *Spatial Transformer* [JSZ+15] module. The localization network predicts the transformation parameters $\theta$ based on the given input feature map U. A regular grid $\varphi$ is defined by the grid generator. The sampling grid $\varphi_\theta$ is applied to U producing the output feature map V. The localization network, grid generator, and sampler build the *Spatial Transformer* module.

as the multi-level scheme (Section 4.3) in classical approaches. The reference image $\mathcal{R}$ is transformed by the *Spatial Transformer* module using the computed transformation $\varphi$. The use of the fully differentiable *Spatial Transformer* allows adjustment of the network parameters with respect to the gradient of the loss function. The authors propose two loss variants: an unsupervised loss based on an image similarity metric and a variant including an additional auxiliary loss function using the Dice score on anatomical segmentations. VoxelMorph is designed for deformable transformations and therefore remains dependent on a stable affine pre-registration step. The VoxelMorph model achieves accuracy comparable to classical methods but is several times faster.

Based on the success of VoxelMorph, other methods have been developed that rely on the use of completely convolutional networks. In [HGH19], the convolutional model is extended using strategies from classical image registration and applied to the challenging task of lung registration. The authors use a multi-level scheme to predict large deformations, which occur during the breathing of the patient between two CT scans. Here a cascaded *UNet* structure is used in combination with a segmentation-based auxiliary loss.

However, for large deformations, existing end-to-end (learn-to-map) models typically struggle to reach the accuracy of classical methods [She+19]. Moreover, these architectures are focused on particular applications and regularly are characterized by their large number of parameters [Hei19].

This has led to the development of new models that no longer solely rely on learn-to-map CNNs but combine classical and learning-based methods. First, individual components of classical optimization methods were replaced by neural networks. The authors of [NKV19] proposed to replace the regularization term $\mathcal{S}$ in (4.1) with a trainable component. Next, models were designed to solve tasks that are not the registration itself but are related to it. In [Hoo+21] a model was trained to perform the — usually very expensive —

process of hyperparameter tuning for a given registration network. Similar to this Mok et al. [MC21b] presented a *conditional registration framework* to optimally chose the regularization weight $\gamma$ in (4.1). The high-dimensional feature maps of a convolutional registration network are "conditioned" on (i.e., depend on) the regularization weight $\gamma$. For this reason, the method is only applicable to CNN-based registration.

While prior work employs these auxiliary strategies to improve the network's performance, the focus of attention has recently shifted to the variational and iterative core of classical methods. These models are more closely related to the model presented in Chapter 5 of this thesis.

**VR-Net.** The *VR-Net* [Jia+21] converts a generic variational problem, such as optimization in image registration, into two sub-problems: a point-wise closed-form solution and a denoising problem. Thereby *VR-Net* embeds

$$\min_{\varphi} \int_{\Omega} \mathcal{D}(\mathcal{R}(x), \mathcal{T}(x+\varphi(x))\, dx + \gamma \mathcal{S}(\varphi(x)) \tag{4.13}$$

into the framework of a neural network. The authors reformulate (4.13), employing the first-order Taylor expansion to derive an alternative problem:

$$\begin{aligned} \mathcal{T}(x+\varphi) &\approx \mathcal{T}(x+\varphi^{\omega}) + \langle \nabla \mathcal{T}(x+\varphi^{\omega}), \varphi - \varphi^{\omega} \rangle \\ \varphi^{\omega+1} &= \arg\min_{\varphi} \int_{\Omega} \mathcal{D}(\rho(\varphi))\, dx + \gamma \mathcal{S}(\varphi), \end{aligned} \tag{4.14}$$

where

$$\rho(\varphi)(x) := \mathcal{T}(x+\varphi^{\omega}) + \langle \nabla \mathcal{T}(x+\varphi^{\omega}), \varphi - \varphi^{\omega} \rangle - \mathcal{R}(x). \tag{4.15}$$

Additionally, they use variable splitting to decouple the data and regularization term

$$\min_{\varphi, v} \int_{\Omega} \mathcal{D}(\rho(\varphi))\, dx + \gamma \mathcal{S}(v) \quad \text{s.t.} \quad \varphi = v \tag{4.16}$$

and add a penalty function to derive the two sub-problems

$$\varphi^{k+1} = \arg\min_{u} \int_{\Omega} \mathcal{D}(\rho(\varphi))\, dx + \frac{\theta}{2} \int_{\Omega} \|v^k - \varphi\|^2\, dx \tag{4.17}$$

and

$$v^{k+1} = \arg\min_{v} \gamma \mathcal{S}(v) + \frac{\theta}{2} \int_{\Omega} \|v - \varphi^{k+1}\|^2\, dx. \tag{4.18}$$

Here (4.18) can be considered a denoising problem for known $\varphi^{k+1}$. Since the Taylor expansion (4.14), holds only true for small deformations $\varphi^{\omega}$, a warping operation is adopted to break large deformations into $N_{\text{warp}}$ smaller steps. Given a closed-form solution of (4.17) the problem in (4.13) can be solved in an iterative manner using $N_{\text{warp}} \times N_{\text{iter}}$ steps. The authors use three custom layers: the *warping layer* (WL) in form of bilinear interpolation, the consistency layer (ICL), imposing intensity consistency, and the generalized denoising layer (GDL), implemented as a residual UNet. The trainable parameters in *VR-Net* consist of the penalty weight $\theta$ in (4.17, 4.18) and the UNet weights. Cascading $N_{\text{warp}}$ of $N_{\text{iter}}$ layers yields the *VR-Net* architecture. The weights can be shared between cascades, reducing the computational cost without a noticeable loss in accuracy. Different initialization strategies for the starting grid $\varphi$ are proposed such as zero, Gaussian distribution, or learned. In the learned variant, another UNet is used to predict the initial grid. The learned initialization works best but increases the overall parameter count due to the additional network. The model is evaluated on 2D and 3D MR data and compared to iterative methods, such as Free Form Deform (FFD), as well as data-driven models, such as VoxelMorph [Bal+19] or RC-Net [Zha+19b]. As metrics, the Dice score and Hausdorff distance on segmentation masks are used. The authors report that their model outperforms both iterative and data-driven methods on the 2D data set and achieves second-best performance behind FFD on the 3D data set. Although VR-Net does not map the images to a deformation directly, its run-time is close to purely data-driven models.

**Multi-scale Neural ODEs.** Some classical methods, such as LDDMM, model the dynamics of the transformation $\varphi$ as a differential equation described by a fixed predefined function. In [Xu+21] a flexible, trainable *ordinary differential equation* (ODE) formulation is used to learn a registration optimizer. Let the function $f(\theta)$ parameterize the transformation $\varphi_\theta$, then the optimization in (4.1) can be written as

$$\hat{\theta} = \arg\min_{\theta} f(\theta), \tag{4.19}$$

where

$$f(\theta) := \mathcal{D}(\mathcal{R}, \mathcal{T} \circ \varphi_\theta) + \gamma \mathcal{S}(\theta). \tag{4.20}$$

Using an iterative solver as in (4.7) and letting the step size approach zero, one arrives at an ODE for the unknown parameters

$$\frac{d}{dt}\theta(t) = f_\omega(\theta(t), t), \quad t \in [0, T]. \tag{4.21}$$

Here $f_\omega$ is a trainable neural network, which enables *Neural ODEs* to use flexible functions in contrast to the fixed predefined versions used in classical methods. In [Xu+21], Runge-Kutta methods are used to solve (4.21) and derive the output at step $K$. Additionally, the ODE is solved at different resolutions yielding a multi-scale ODE model. The model consists of three scales each executing three update steps ($K = 3$). An unsupervised loss composed of a similarity metric, a perturbation-based self-similarity, and regularization is used. The similarity metric is computed on extracted context vectors and the authors also design a "disentanglement" network to deal with different images from different modalities. For each image, a context and style representation is created, which is used by a generator network to produce images of the "same" modality. The *Neural ODE* model addresses the problem of large deformations by cascading an affine and deformable variant of its architecture to build a hybrid model. For evaluation, the BraTS MR

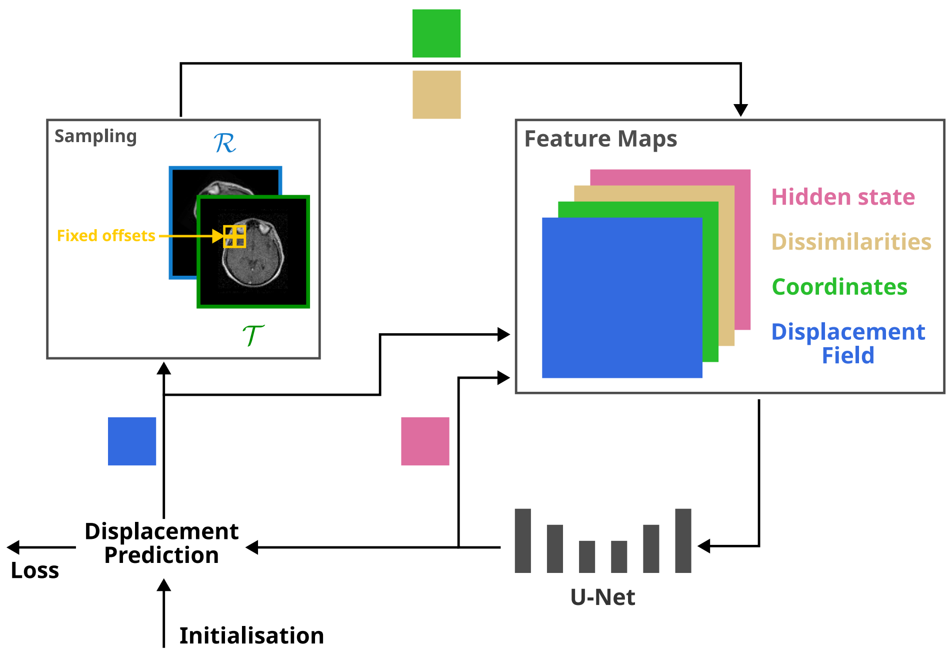


Figure 4.5.: The *L2O* [FHH22] optimizer network. The input to the model is a four-fold feature map. It consists of the preceding displacement, possible displacements for every voxel in the form of coordinates, the dissimilarity cost of each sampled displacement, and hidden states comprised of output feature channels from the UNet backbone. The authors apply eight recurrent iterations of the here depicted model to build a trainable optimizer.

data set is used. Large deformations are created by simulating affine and deformable transformations, in a similar manner to the data creation in Section 5.3 of Chapter 5. The *MS-ODE* outperforms all tested iterative and learning models based on the Dice score and root mean square error on the simulated deformations.

**Learning Iterative Optimization.** In [FHH22], the *L2O* model is proposed, a network designed to mimic the behavior of the popular Adam optimizer. The model consists of a UNet backbone, which is recurrently applied to adopt an iterative update scheme. In contrast to the *MS-ODE* model, it relies on pre-registration to handle large deformations. The composition of the inputs is particularly interesting, which consists of displacement coordinates, a dissimilarity cost calculated on MIND [Hei+12] features, and a hidden state propagated through the recurrent iterations of the network. The *L2O* optimizer network and its inputs are depicted in Figure 4.5. The authors point out that the image pair is not provided directly as an input to the model, mirroring the relationship of input and model aimed for in our network design (Chapter 5). Another similarity to our approach is the use of hidden states to provide information containing nonlinear relations between steps. Here selected channels of the UNet's output feature map are used as hidden states, while in our model, the hidden state of an LSTM layer serves a similar purpose. The authors apply eight recurrent iterations through the backbone UNet in order to build an optimizer. To favor fast convergence of the network optimizer, deep supervision is used. In deep supervision, the loss is calculated using the outputs of each level of the network. Here a level can denote a layer in the network or a step in a recurrent application. This type of loss calculation increases the influence on the initial iterations in a recurrent scheme. The *L2O* model is trained on multiple registration tasks, including the EMPIRE or DIR-Lab COPD data sets, and evaluated using the target registration error on 300 corresponding landmarks. The authors report superior results to the base-line Adam in the case of none pre-registered image pairs while yielding lower accuracy on pre-registered images. The *L2O* model produces smooth deformations, indicated by the low Jacobian determinants, and converges considerably faster than Adam.

All of the latter network models use concepts of variational optimization to combine the accuracy of classical methods with the speed of learning-based image registration. In a similar spirit, we focus on using a rough network-based estimate to guide classical methods in Chapter 5.

# 5. Meta-Learning for Image Registration

Classical registration methods often rely on a coarse alignment to produce optimal results [Kön18]. As early as 2016, a proposal was made in [Lia+17] to use a trainable agent for this purpose. A first step is to combine classical registration with learning processes. In this chapter, we will expand on that idea by introducing concepts from the field of meta-learning to classical image registration methods. We focus on large deformations to increase the performance of simple structured optimizers (Section 5.4).

Most models in the field of image registration can either be attributed to classical methods, such as variational approaches, or more recent network-based models. We aim to develop a framework incorporating strategies of both approaches. The resulting model is presented in Section 5.2. We start by providing a brief overview of existing network-based registration and popular meta-learning strategies (Section 5.1). The model architecture is the main contribution of this thesis and is presented in detail in Section 5.2. As we have adopted a supervised training approach, we need artificial ground truth deformations, the creation of which is described in Section 5.3.

We test our model design on multiple data sets and benchmark our approach against the established *FAIR* and *elastix* packages. The results for the *affine* and *non-parametric* deformation settings are provided in Sections 5.4, 5.5. Finally, in Section 5.6 we discuss the results and place our model in the context of existing registration models.

## 5.1. Related Work

The growing success of neural networks in a diverse set of machine learning tasks, such as image recognition [He+16] and translation [Vas+17], motivated their application to the task of image registration. One approach is to replace individual parts of classical processes. For example, in [Yan+17] a network is used to predict the initial momentum of LDDMM. Far more frequently, neural networks are used in an end-to-end fashion. The design can vary greatly, but all methods have in common that they learn to directly map one image to another, an example is the VoxelMorph architecture [Bal+19]. For an overview of other influential representatives, see Section 4.5.

The approach in this thesis is inspired by meta-learning, which can increase the effectiveness of existing optimization methods through data-driven learning: Already in 2001, Hochreiter et al. [HYC01] proved that Recurrent Neural Networks can be used to form *efficiently trainable* optimizers that can adapt to a variety of problem classes. Building on this, Andrychowicz et al. [And+16] showed that trainable optimizers can achieve better results — measured in terms of the loss — than manually designed update rules, including Stochastic Gradient Descent, RMSprop, Adam, and Nesterov's Accelerated Gradient Descent.

Due to their strong generalization capabilities, meta-learning networks can also be used to solve the most common computer vision tasks, as demonstrated in [FAL17]. In a more general way, these networks can be trained to approximate any updating operator [AÖ17]. Given an objective, the trained operator calculates gradient-like information while being embedded in a classical gradient descent optimization. This approach yields remarkable results in reconstructing images from projections of simulated CT images.

The methods in [Hoo+21; MC21b] aim to reduce user interaction by proposing trainable strategies for the hyperparameter search. In a similar fashion, our affine model is partly designed to produce robust starting points for classical methods, reducing outliers. In a classical setting, these would have to be manually corrected.

In [MC21a], Mok and Chung expand on their work [MC21b] to further accelerate hyperparameter tuning, replacing all residual blocks in the *Deep Laplacian Pyramid Image Registration Network* (LapIRN) [MC20] with a conditional instance normalization module. The cLapIRN achieves top-performing results on multiple tasks in the Learn2Reg 2021 challenge [Her+22].

Some recent work tries to bridge the gap between traditional iterative and learning-based methods. Important representatives with methodological affinities to the model proposed in this thesis have already been described in more detail in Section 4.5. In line with that work is the *Gradient Descent Network for Image Registration* (GraDIRN) [Qiu+22]. The model embeds an unrolled multi-resolution optimization and calculates iterative update steps based on image dissimilarity. In the fashion of a bi-level optimization, the inner level aims to align two images, while the outer level uses a self-supervised loss to update the network's parameters. This method has conceptual similarities with our approach. An iterative methodology is used and the approach is designed as a bi-level optimization. The method is fully learning-based and requires upstream affine registration. Furthermore, the network is not used to solve large deformations as we intend with our model.

The *Recurrent Registration Neural Network* (R2N2) [San+19] replaces the set of fixed basis functions in parameterized image transformations with trainable Gaussian kernels to align an image pair. A local deformation is defined as a Gaussian function with trainable position, shape, and weight components. The model consists of three sub-networks: a convolutional variant of the gate recurrent unit (GRU) named *gated recurrent registration unit* (GR2U), an extension of the LSTM layer, a network to predict the position of the basis function, and finally a network to predict the parameters for the Gaussian. Evaluated on 2D lung MR images, the *R2N2* achieves comparable results to a baseline B-spline registration while gaining a speedup of factor 15. The sequential registration of this approach is similar to our iterative method. However, the focus is only on small local deformations that can be corrected by B-spline interpolation. As the model predicts the parameters of the B-spline functions, they are bound to the form of Gaussian functions. Furthermore, a *Spatial Transformer* module (Section 4.5) is required to apply the displacement to the images. Our approach, on the other hand, directly generates a dense displacement field and allows its application without the use of the *Spatial Transformer* module.

Conceptually, our model is most similar to [FHH22; Qiu+22] in that its core is an iterative update scheme parameterized by a neural network. Our approach differs in that we are primarily interested in solving large deformations. Therefore, our model serves to provide initial values for a classical method and, unlike [FHH22; Qiu+22], does not require an upstream registration by another network.

## 5.2. Registration Model

We aim to incorporate the formulation of an iterative update of classical, gradient-based optimization

$$x_{k+1} = x_k + \alpha_k \nabla f(x_k) \tag{5.1}$$

into a neural network setting. The step size $\alpha_k$ is usually set using a line search algorithm, such as the Armijo rule (Section 4.3). The structure corresponds to a bi-level optimization problem [Fra+18] and also exists in a similar form in the field of meta-learning [FAL17] or [Hos+21] for a comprehensive overview.

We retain the iterative nature of classical registration approaches but replace the update step with a neural network in a similar way to meta-learning. The network $\Theta_\gamma$ then encodes a nonlinear update:

$$x_{k+1} = x_k + \theta_k, \quad \theta_k := \Theta_\gamma(\nabla f(x_k)). \tag{5.2}$$

The minimization process can now be automatically tuned to a specific problem class by adapting the weights in $\Theta_\gamma$, in contrast to relying on generic strategies such as Newton- or Quasi-Newton methods.

Compared to the construction of a classical image registration model (4.1) from Section 4.1 the choice of the objective $f$ is less crucial. Classically, the quality of the obtained deformation is directly affected by the choice of the data term $\mathcal{D}$ and regularizer $\mathcal{S}$ that make up the objective function $f$. In our model, the choice of $f$ is less important, because it only serves as a *guide* to construct an iterative process of the form (5.2). Like any other neuronal network, it can be trained using any loss and is not bound to finding a good minimizer of a given objective. Here, we use a simple SSD term $\mathcal{D}(R, T) := \|R - T\|_2^2$.

To build a model that is at least as powerful as an iterative method with $K$ iterations, we repeat (5.2) $K$ times by stacking a corresponding number of layers. As an alternative, we investigate whether $K$ iterations through layers with shared weights are capable of performing similarly. The model created in this manner should benefit from the non-linear gradient information on $f$, thus adding the potential of non-linear, trainable steps. Comparable to the method presented in [FHH22], the model has no direct access to the input images $\mathcal{R}$ and $\mathcal{T}$ and information about the problem is exclusively passed through the gradient of the distance function $f$.

**Model Graph.** We prepend a *projection layer* $L_{\text{in}}$ to our *main building block* $\Theta_\gamma$. This enables us to alter the dimension and allows to either reduce or expand the *main block's* feature size. In addition, it allows for the combination of input channels, hence we call this layer *Fuse Layer*.

The update (5.2) is implemented employing an LSTM cell, following [HS97]. We decided to use the LSTM cell because it features a *hidden state* that allows us to carry over additional history from previous layers. In this way, in our model, the update not only depends on the current iterate $x_k$ and gradient $\nabla f(x_k)$ but also on the hidden state $h_k$ that can potentially incorporate previous gradient and function value information. The LSTM output $o_k$ is then passed to the another *projection layer*, the *Scale Layer*, $L_{\text{out}}$:

$$
\begin{aligned}
(h_{k+1}, o_k) &= \text{LSTM}(h_k, L_{\text{in}}(\nabla f(x_k))) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_h} && (5.3)\\
\theta_k &= L_{\text{out}}(o_k) \in \mathbb{R}^{d_h} \times \mathbb{R}^{d_x} \ , && (5.4)\\
x_{k+1} &= x_k + \theta_k, && (5.5)
\end{aligned}
$$

where $d_x$ is the dimension of transformation parameters, and $d_h$ is the size of the LSTM hidden states. The *Scale Layer* acts as the counterpart to *Fuse Layer* and projects the outputs of the LSTM layer back into the space of the inputs. Our complete network is of the form

$$\Theta_{\gamma,K} = L_{\text{out}}(\text{LSTM}_k( \quad \dots \quad \text{LSTM}_0(h_0, L_{\text{in}}(\nabla f(x_0)))))). \qquad (5.6)$$

This theoretically allows our model to mirror successful methods with gradient history such as L-BFGS [NW06, Chap.6]. As inputs, we only use gradient information, derived from the distance function $f$. The images $\mathcal{R}$ and $\mathcal{T}$ are used solely to calculate the new distance after each layer update.

Figure 5.1 shows the model graph for two repetitions of a full network. The LSTM cells have individual weights, whereas the *Fuse Layer* $L_{\text{in}}$ and *Scale Layer* $L_{\text{out}}$ share their respective weights over the $K$ repetitions. Model weights are initialized to implement an approximate identity mapping in order to avoid that early updates are directed against the descent direction.

We introduce two types of *projection layers* in our model: the *Fuse*- and *Scale Layer*. These two layers project the inputs to the model feature space or the outputs to the space of transformations. The input to our model can either be the gradient of the distance measure, the regularizer, or a combination thereof. Generally, the gradient is calculated with respect to each transformation parameter. Consequently, the dimension of the input can be rather low, in the case of affine transformations, or considerably large when dealing with deformable transformations. Additionally, if we want to incorporate the gradient of both, the distance measure and the regularizer, the multichannel input has to be reduced to fit the single-dimensional LSTM design. The *Fuse Layer* design is illustrated in Figure 5.2.

The *Fuse Layer* is designed to reshape the input to fit the main module and allows to change the input dimensions. In the case of affine transformations, a linear layer achieves the best results. Non-parametric transformations are implemented as a vector

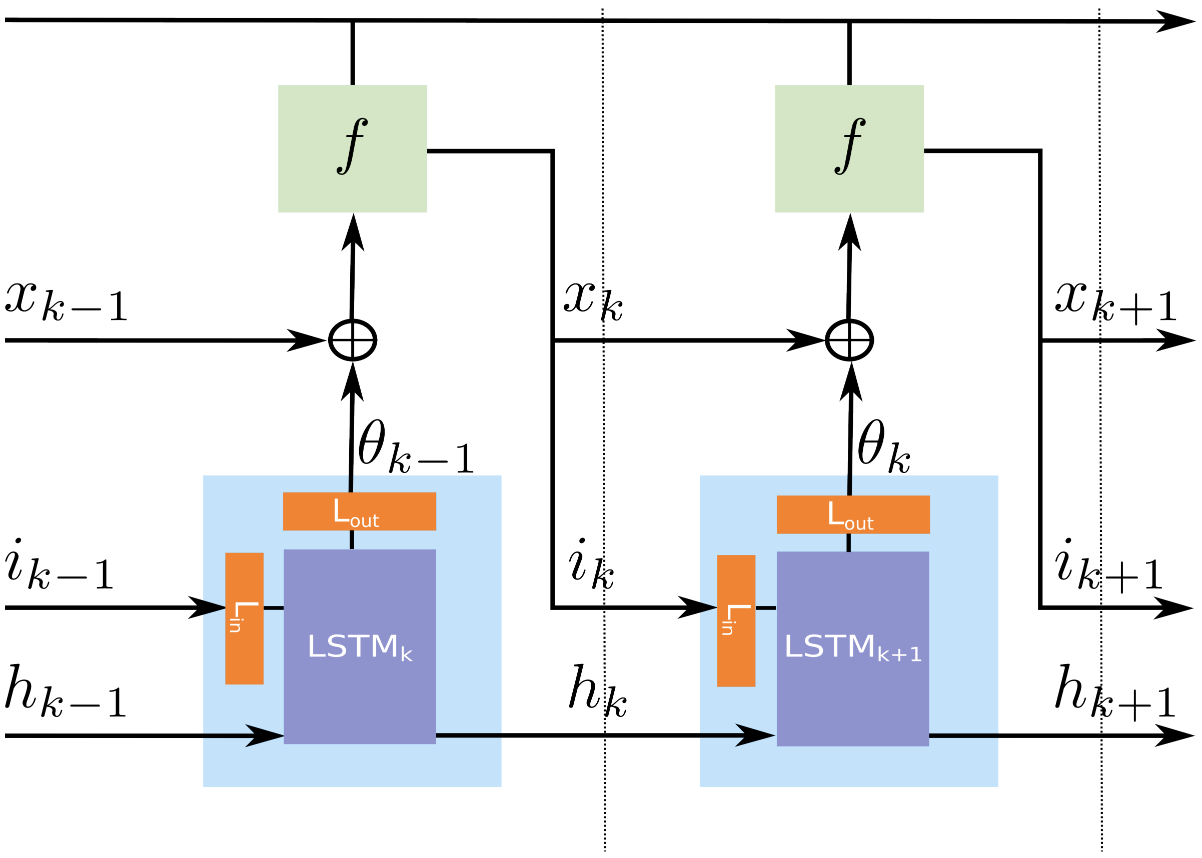


Figure 5.1.: Computational graph for constructing the registration network. The inputs $i_{k-1}$ are passed through the *Fuse Layer* ($L_{\text{in}}$) reducing their size and fusing the input channels. The main block $\Theta_\gamma$ predicts the update $\theta_k$ in (5.2) and new hidden state $h_k$ from the nonlinear gradient information $i_{k-1} := \nabla f(x_{k-1})$ and the previous hidden state $h_{k-1}$. The *Scale Layer* ($L_{\text{out}}$) restores the original size and creates the update $\theta_k$. The update is then added to the iterate $x_k$. Repeating this process $K$ times, the resulting stacked architecture imitates the structure of classical iterative optimization methods, while providing the necessary degrees of freedom to make the process more efficient and robust through training.

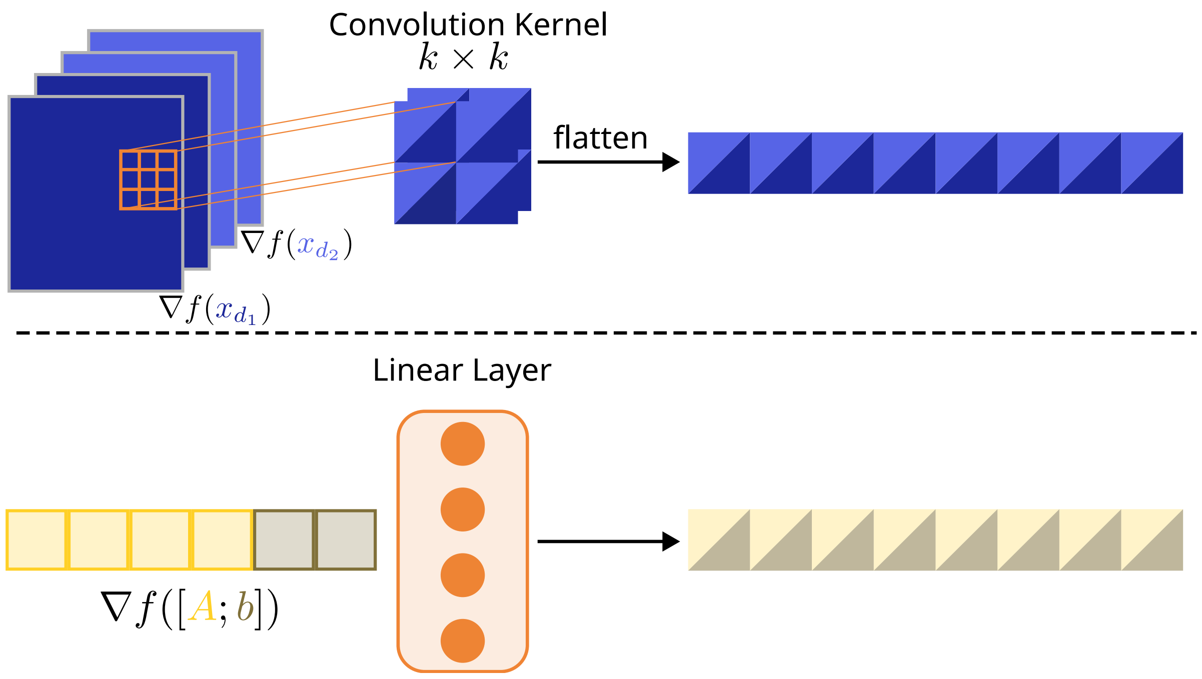


Figure 5.2.: Variants of the *Fuse Layer* (orange): convolutional and linear. **Top**: A convolution with kernel size $k$, here $k = 3$ is depicted. The convolutional *Fuse Layer* is applied to the deformation grid. In the case of a two-dimensional image ($d = 2$), we aim to combine the gradients of the displacements in all image directions. Therefore, the gradients for the two directions ($\nabla f(x_{d_1})$, $\nabla f(x_{d_2})$) are processed as different channels (different shaded blue squares). The channels are reduced and the gradient information from both directions is combined (mixed blue squares). Afterwards the spatial dimensions are reduced to one, using a flattening operation. **Bottom**: Affine transformations are parameterized by a rotation matrix $A$ (yellow) and translation vector $b$ (brown), see also Figure 5.4. Passing the gradients $\nabla f([A; b])$ directly into the LSTM layers limits feature space within the model. We project the inputs onto a higher dimensional feature space by expanding its size using a linear layer. Similar to the convolutional variant the gradient information of rotation and translation is combined during the process (yellow-brown squares).

field of point-wise deformations. Therefore, the input is a two-dimensional grid and a convolutional layer of fixed weights is used. We also evaluated trainable weights but found that these made training more difficult. The weights are set to implement a mean filter over the spatial dimensions.

We have established that the inputs must be adapted to the sequential design of the LSTM layer. The output of the main block must now be projected back into the space of the transformations. This is the task of the *Scale Layer*. Here there are again several versions according to the Fuse Layer. For affine transformations a linear layer is sufficient. For the case of non-parametric transformations, we test different designs from simple bilinear interpolation to fully trainable convolutional layers, which we discuss in Section 5.5.

To stabilize the training of our network, we normalize the input by dividing it by its 2-norm:

$$\nabla\overline{f(x_k)} = \frac{\nabla f(x_k)}{||\nabla f(x_k)||_2 + \epsilon}. \tag{5.7}$$

We set $\epsilon$ to $10^{-8}$ throughout all experiments.

### 5.2.1. Loss

Since no natural ground truth exists in image registration, self-supervised or weakly-supervised approaches [HGH19; MC21b] are often used in practice. In contrast, we employ a supervised learning method by using simulated deformations to create a ground truth, similar to [Dos+15]. We specifically do *not* use the distance $f$ as loss, thus providing the network with the opportunity to freely find the optimal registration in terms of *distance to the ground truth deformation map.*

**Deform Loss.** In this way, we enable the network model to not necessarily construct a better solver for minimizing $f$ but to pursue the underlying primary goal of finding a good registration. The solution is not limited to the quality of iterative methods and allows for a more forgiving choice of the distance and regularizer used in $f$. We define the loss directly by comparing the mean squared error (MSE) of the deformation fields:

$$\mathcal{L}_D(\varphi_{x_K}) = \frac{1}{n}\sum_{i=1}^{n} ||\varphi_{x_K}(z_i) - \varphi_{\bar{x}}(z_i)||_2^2 \tag{5.8}$$

where $\{z_i|i \in \{1, \ldots, n\}\}$ is the image grid, $\bar{x}$ denotes the ground truth deformation parameters and $x_K$ is the output of $K$ applications of $\Theta_\gamma$ as outlined above. We call this the *Deform Loss.*

**Trajectory Deform Loss.** Other work has shown that *deep supervision* [FHH22; LeC+21] increases accuracy and enables faster convergence of neural networks for tasks such as registration or classification. The idea is to add loss terms based on the activations of the intermediate layers, e.g., at each scale in a multi-level scheme [LeC+21], or at each iteration [FHH22]. Applying this idea to (5.8), we propose to use

$$\mathcal{L}_T(\varphi_{x_K}) = \frac{k}{K} \sum_{k=1}^{K} \mathcal{L}_D(\varphi_{x_k}) \tag{5.9}$$

where $k \in \{1, \ldots, K\}$ denotes the step at which the *Deform Loss* $\mathcal{L}_D(\varphi_{x_k})$ is calculated. This *Trajectory Deform Loss* is designed to enable the initial steps to influence the loss computation and avoid early deformations that are completely unrealistic while maintaining the focus on the last step.

**Multi-Level Loss.** As discussed in Chapter 4, classical methods benefit strongly from a multi-level approach. Similarly, we consider a multi-level variant of our network. Along with this model, we test a version of the loss function that is applied at each resolution level. Starting from (5.8), we denote the final transformation $\varphi_{x_K}$ of each level $l$ as $\varphi_l$ and the final transformation of the whole model as $\bar{\varphi}$. Then the *Multi-Level Loss* is defined as

$$\mathcal{L}_{ML}(\bar{\varphi}) = \frac{1}{L} \sum_{l=1}^{L} \mathcal{L}_D(\varphi_l). \tag{5.10}$$

Both *Trajectory Deform Loss* $\mathcal{L}_T$ and *Multi-Level Loss* $\mathcal{L}_{ML}$ operate utilizing the *Deform Loss* $\mathcal{L}_D$ on either multiple steps or resolutions and therefore could be considered a form of deep supervision. They are designed to encourage faster convergence in the early stages of the iterative process at the center of our model design. In theory, it is also possible to replace $\mathcal{L}_D$ with $\mathcal{L}_T$ in (5.10), which we did not investigate in the context of this thesis.

### 5.2.2. Scheduling

In classical optimization, the choice of appropriate step size is crucial. Therefore, a line search method, such as Armijo (Section 4.3), is used to determine an optimal step size. In the field of neural networks, the choice is often less important, since one is generally more interested in a sufficient and robust reduction of the loss than in convergence to a possible (local) minimizer. A working step size is often determined via parameter search and fixed for the time of training. Step size is the notation favored in the optimization community, the learning community denotes it as learning rate, but the two terms can be used interchangeably. For the sake of consistency, we will adopt the notation of learning rate from here on.

Adjusting the learning rate during training can promote model convergence. Small learning rates result in more updates since the changes applied to model parameters are small. On the contrary, too large learning rates may lead to divergent behavior. Since in this work we are designing a model that has an iterative update structure, several scheduling rules are tested.

The first scheduling rule is *Reduce On Plateau*, which reduces the learning rate by an assigned factor once learning stagnates. The *StepLR* variant reduces the learning rate using a multiplicative decay factor after a defined period of training epochs. The third evaluated scheduler is *Exponential Decay*, which decays the learning rate by a fixed factor every epoch. Another technique is proposed in [LH17], the *Cosine Annealing*:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{T_{\text{cur}}}{T_{\max}}\pi\right)\right), \tag{5.11}$$

where $\eta_{\max}$ is set to the initial learning rate and $\eta_{\min}$ is a lower bound.

### 5.2.3. Gradient Calculation

The model proposed in this thesis can be viewed as a bi-level optimization problem. The inner level is the registration of the images by adjusting the transformation parameters $x_k$, aiming at the reduction of the distance between the deformed images, see (4.1) in Section 4.1. The outer level consists in training the network itself to derive the update $\theta_k$ (5.2) for the inner level. Algorithm 1 depicts the wrapping algorithm used to embed the inner optimization into the network.

**Algorithm 1** Wrapping function to enable batch-wise gradient calculation using `autograd`. The inner level cost function can be arbitrary. For image registration, we use the definition in (4.1).

```
Input: $x_k$: iterate at step $k$
Input: $\mathcal{T}, \mathcal{R}$: template and reference image
Input: $f$: inner level cost function
1: $x_k \leftarrow x_k$.requires_grad_(True)      ▷ Enable graph-based gradient calculation
2: $f_k \leftarrow f(x_k, \mathcal{T}, \mathcal{R})$
3: $\nabla f(x_k) \leftarrow$ autograd.grad($x_k, f_k$)
4: return $\nabla f(x_k)$
```

The implementation of the inner level is of utmost importance, as the gradient $\nabla f$ is used as input for the network. This results in two requirements for the implementation: the inner level has to be differentiable to derive the gradient at each step and the calculation of the gradient should be as precise as possible. We want to use the outputs as initial values for a classical gradient-based method that requires such a high degree of accuracy. In this way, we ensure a smooth interaction between the two methods. The latter is realized by using a double-precision floating-point format instead of single-precision. The disadvantage is that the network model itself and especially the inputs of the inner level also have double-precision format, which increases memory consumption. In the spirit of the implementation presented in *FAIR* [Mod09], the inner optimization is fully differentiable and gradient calculation is performed by employing the `autograd` routine provided by the *PyTorch* framework. We wrap the `autograd.grad` subroutine to enable batch-wise gradient calculation. We pass the iterates $x_k$ of step $k$ in the inner

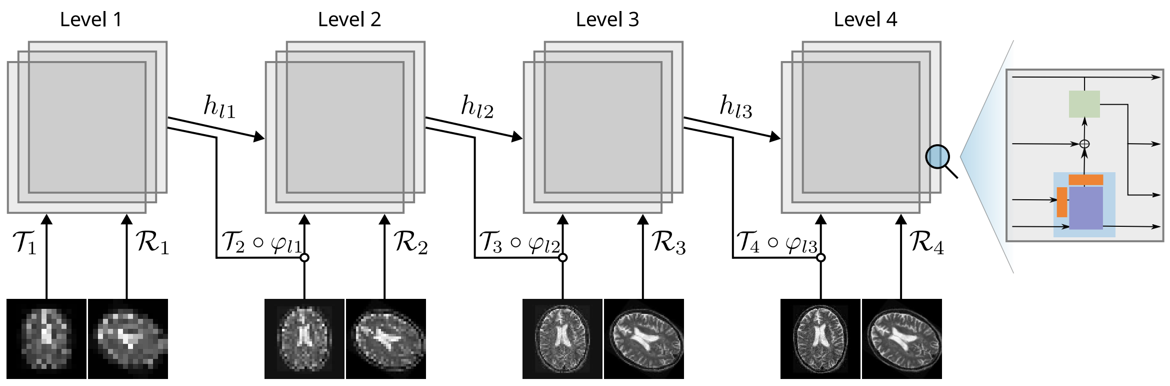


Figure 5.3.: Multi-level variant of the proposed model. Following the classical multi-level scheme, the image pairs are passed to sub-networks with increasingly higher resolution. Each of these sub-networks contains $K$ repetitions of the *main building block* (Figure 5.1) as well as the weight-shared *Fuse*- and *Scale Layers* (Figure 5.2). Each sub-network returns its final transformation $\varphi_{l_i}$ to deform the next level template image $\mathcal{T}_{i+1}$ before it is passed into the next sub-network. Moreover, the sub-networks hidden state $h_{l_i}$ is also passed to the following block to enable the flow of high-level gradient information throughout the complete multi-level model.

level as inputs and calculate the outputs using $f$ in (4.1). This allows us to calculate $\nabla f$ for multiple image pairs at once with respect to $x_k$.

### 5.2.4. Advanced Strategies

**Multi-Level Models.** Classical iterative methods use application-specific strategies to elevate their performance. Most notable is the multi-level scheme to avoid local minima and to speed up runtimes by using the low-cost coarse results as initialization for the finer resolutions. This strategy is so effective that many learning-based approaches also use it to improve their accuracy [HGH19; Xu+21; San+19; MC20]. Due to this undeniable advantage, we have implemented a multi-level variant of our model. Figure 5.3 illustrates the multi-level model. The *multi-level model* consists of $L$ sub-networks that operate on different resolutions of the image pair. The template image $\mathcal{T}_{l+1}$ is transformed with the predicted $\varphi_l$ of the previous level before it is passed to the next sub-network $l+1$. The hidden states $h_l$ are also passed to the next level to keep track of past updates.

**Freeze Strategies.** Generally, all layers of a model are trained simultaneously. For our iterative model, this would mean to train all $K$ update steps at the same time. However, the quality of the first steps is crucial for the calculation of later steps. Therefore, it makes sense to train the steps in isolation first, in order to ensure that the first steps produce meaningful updates. We do this by progressively training each of the main blocks in our model. Each block is trained for a limited number of epochs, then the weights are fixed and the process continues with the next block until each layer has been trained. We evaluate different *freeze strategies*: either keeping past layer weights fixed or allowing for combined training of all preceding layers, adding successive layers once a required number of epochs is met. A similar approach is used in [HGH19], where the authors train each level of their multi-level *UNet* in the same progressive manner.

## 5.3. Data

In image registration, the ground truth is usually not known. Traditional approaches do not need a ground truth, as they do not *learn* from data, and instead rely on hand-crafted distance functions and regularization [Mod03]. Learning-based methods typically work in an unsupervised [Vos+17; Zha+19b; Jia+21] or semi-supervised fashion [HGH19]. As a result, the formulation of the loss function is heavily studied, often employing support functions to encode the desired property [HGH19]. This may lead to loss formulations heavily adapted to fit a given task.

Our approach is inspired by the work in [Dos+15], where the authors solve optical flow estimation as a supervised learning task. To generate the large Flying Chairs data set, the authors randomly sample 2D affine transformation parameters for the background and chair objects. In creating their own translations, the authors obtain a set of image pairs, ground truth optical flow, and occlusion regions.

In [Dos+15], the resulting data set is of fixed size and data augmentation is used to further increase data variability. Ideally, our network does not encounter a deformation twice during training for optimal generalization capability. Therefore, we use an “on the fly” routine, drawing samples from the given deformation distribution during training. Consequently, we do not have a fixed training set. However, the validation and test set are created once and then fixed — i.e., saved to disk — in order to enable comparison of model performance. To ensure reproducibility, we pass a random number generator to the data creator and attribute a starting seed point to every training, validation, and test set.

**Affine Deformations.** For the *affine deformations*, we create an artificial data set by deforming a $128 \times 128$ template image, keeping the original image with added noise as a reference and the deformed variants as templates. Transformation parameters are chosen randomly from a uniform distribution of rotations in $[-60°, 60°]$ and translations of at most a quarter of the image dimensions. The shear and scale factors are defined as entry-wise manipulations of the rotation matrix $A$. The size of the manipulations is also randomly chosen up to 15% of the maximum rotation angle. In Figure 5.4 the data creation process is visualized. The creation of affine deformations is straightforward. The limited number of degrees of freedom and formulation as matrix multiplication enables us to synthesize image pairs. We can create reasonable deformations by choosing suitable limits of the underlying uniform distributions. To be able to utilize the same supervised learning strategy for *non-parametric transformations* poses a problem. As the name

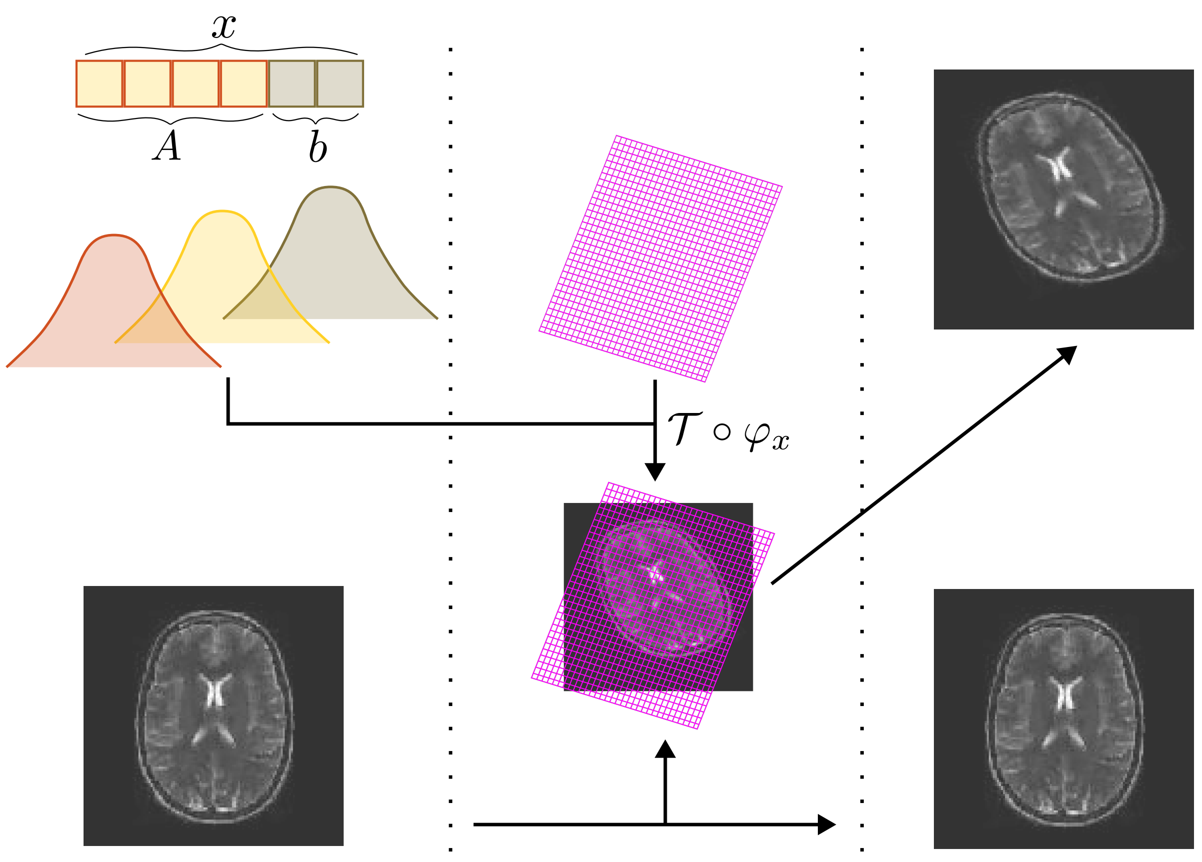


Figure 5.4.: Creation of image pairs to use as ground truth, affine case. **Left**: Affine deformations can be encoded by a rotation matrix $A$ and a translation vector $b$ combined in a transformation vector $x$ with 6 degrees of freedom. We create $x$ using three distinct uniform distributions. First, we draw a rotation angle as a base value for the first four entries of $x$. Second, four shear values are drawn from the second distribution and added independently to the different rotation matrix $A$ entries. The translation vector $b$ consists of two random values. **Center**: Applied to a regular grid, the transformation produces a deformed grid. Transforming the image with the given grid using interpolation results in a new image. **Right**: Keeping the original image as a template and the deformed image as a reference yields an image pair. The transformation $\varphi$ between these images is known, thus enabling supervised learning.

suggests we no longer have a limited set of parameters to control the movement globally. Instead, each grid point can theoretically be freely moved to every other position in the image.

**Non-parametric Deformations.** In reality, those movements are confined by physical constraints such as tissue density or object rigidness, and non-parametric deformations are characterized by strong local and less global movement. We model this behavior through a combination of randomly sampled supporting points and a linear combination of *Radial Basis Functions* (RBF). The following description of randomly generating these deformations is best viewed in conjunction with Figure 5.5:

1. The input image is **downscaled** by a chosen factor. We achieved the best results by setting it to 2, i.e., an image of size 128 is reduced to $64 \times 64$. This is necessary to expand the effect of the intended deformation beyond small neighborhoods around the supporting points.

2. We want to avoid changes in the background. As these are not accounted for in the objective function $f$, see Chapter 4. Background pixels often have the same intensity, so local shifts in this region do not result in a changed SSD value. Consequently, the gradients passed to our model do not change either while the loss on the deformation grid is affected by background changes. This unnecessarily complicates learning for our model. We avoid these deformations by determination of the object using a simple detection algorithm, which uses a rectangular box to define the object.

3. We **randomly choose coordinates** within the limits of the detected object. Again we introduce a parameter to control the number of coordinates, we want to transform. The parameter defines which fraction of coordinates will be shifted. Multiple fractions are tested and we report the corresponding value for each data set.

4. In a similar manner we choose some coordinates outside the object to be fixed. As the grid is created using interpolation, we need some coordinates outside the object to stay the same. Otherwise, the interpolation would create deformations outside the object. Here we select every sixth coordinate, which is marked as red crosses on the left side of Figure 5.5.

5. The coordinates in step (3) are shifted independently in all image dimensions. We apply a **random translation** to the supporting points, marked by blue stars in Figure 5.5. The translation limits are determined by $h \cdot s$. Here $h$ denotes the cell spacing, introduced in (4.3) in Section 4.2, and $s \in [0, 1]$.

6. Both the shifted coordinates (inside) and fixed coordinates (outside) are passed to a **RBF interpolation** as supporting points. In this thesis, we make use of the RBF interpolation function provided by the `scipy package` (1.2.1) [Vir+20]. We

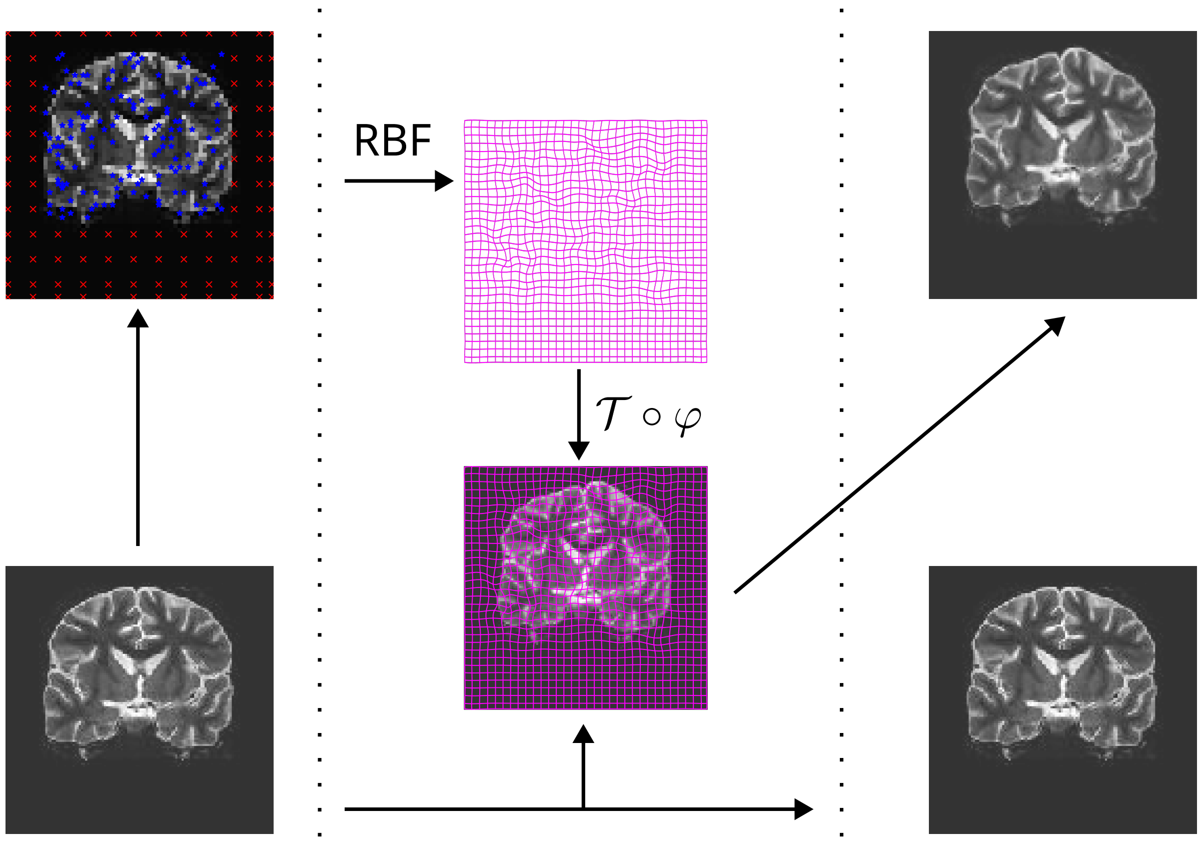


Figure 5.5.: Data set generation for non-parametric deformations. **Left**: The image is scaled down (1), by a factor of 2, and the object is automatically detected (2). The object boundaries define the intervals from which support points are randomly extracted (3, marked by blue stars). To avoid strong deformations in the background, coordinates are selected at regular intervals with a stride of 6 (4, marked by red crosses). **Center**: The support points from step (3) are shifted randomly in all image directions (5). The shifted coordinates in the interior and fixed coordinates in the exterior of the object are passed to the RBF interpolator. The resulting grid is used to transform the original image (6) and Gaussian noise is added. **Right**: The original image is used as template $\mathcal{T}$ and the transformed image as reference $\mathcal{R}$.

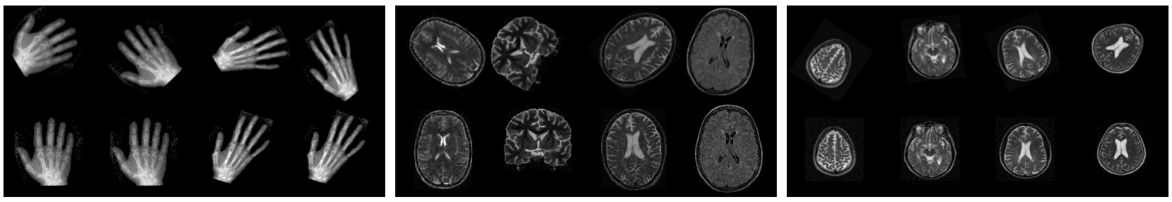


Figure 5.6.: Sample image pairs from *Hands* (**left**), *Brain* (**center**), and *fastMRI* (**right**).

employ thin-plate basis functions $r^2 \odot \log(r)$ for the *Blobs* data set and $\sqrt{\left(\frac{r}{\epsilon}\right)^2 + 1}$ otherwise, here $r$ denotes radius. The result of the interpolation is a deformed grid, which is used to deform the input image from step (1). We keep the original image as template $\mathcal{T}$, add Gaussian noise to the transformed image, and use the final result as reference $R$ for the registration task.

**Data Sets.** Both data generation methods enable us to use a supervised learning approach, as well as define increasingly large deformations. We evaluated the models on three registration tasks:

- A simple synthetic example of registering x-ray images of two different hands *(Hands)*,
- an MRI *Brain* data set from the *kaggle* platform originally designed for tumor detection, containing samples with and without tumor regions [1],
- and DICOM brain images *(fastMRI)* from the NYU fastMRI data set [Zbo+18; Kno+20].

Registration tasks are created by applying the random transformation, as described above, to a randomly selected image. Exemplary image pairs for affine deformations are displayed in Figure 5.6.

[1]https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection — visited 2021

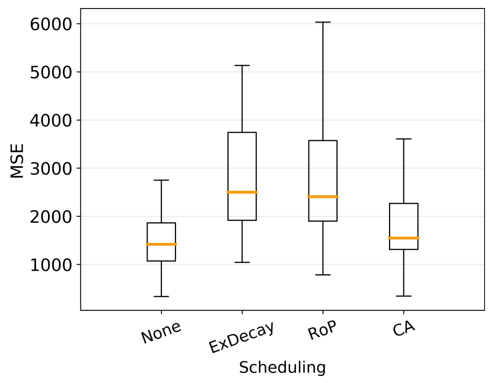
## 5.4. Affine Image Registration

We apply our newly designed model to the task of *affine image registration*, where the deformations can be described by a matrix multiplication and a translation, taking the form $\varphi_x(z) := Az + b$ with $A \in \mathbb{R}^{2\times 2}$ and $b \in \mathbb{R}^2$. As a result, the iterates $x_k$ consist of the unknown rotation matrix $A$ and translation vector $b$, as displayed in Figure 5.4. We omit the regularizer, i.e., $\mathcal{S} = 0$, as the restrictions inherent to affine deformations provide sufficient regularization. We address the problem of large deformations and propose a non-linear, trainable registration step to provide robust starting points for subsequent classical methods.

### 5.4.1. Experiments and Results

**Scheduling.** The updates in earlier steps are usually large, whereas later updates progressively become smaller in size. For this reason, we also address the question of whether a flat network with split weights is able to approximate the early and late update steps. We also investigate the influence of various scheduling techniques on our model with shared weights. We focus on a dynamic adjustment of the learning rate for the outer optimization. The model configurations here are as follows: the initial learning rate is set to $10^{-3}$, the batch size is 50, and we test 10 iterative passes through the model. A layer consists of a main building block of our architecture (see Figure 5.1).

We train the network on synthetic x-ray hand images. The deformations are created as described in Section 5.3, using a maximum angle of 60 degrees, no shear, and a maximum translation of one-tenth of the image dimensions in all spatial directions. The performance is evaluated on a separate test set, created with the same transformation parameters but a different seed point for the random number generator. We evaluate the quality of the registration using the $L_1$ distance of the deformation grids. Three different scheduling techniques and a model without any scheduling are compared. The three techniques are exponential decay (ExDecay), reduce on plateau (RoP), and cosine annealing (CA). The number of epochs varies between 150 epochs for no scheduling and 300 epochs for RoP, due to the employment of early stopping, which terminates the training if the loss no longer declines. Without scheduling the model achieves the lowest average $L_1$ distance of 1522 followed by CA with an average of 1828. Both ExDecay and RoP express higher average errors nearly doubling the distance to 2774 and 2759. On the right side of Figure 5.7, registration results for the model with no learning rate scheduling can be seen. The model often gets stuck in local minima between the individual fingers and scheduling does not improve performance. Multiple iterative applications of a single main building block are insufficient compared to a classical method and scheduling does not improve the model performance. Furthermore, a shared weights model cannot map large and small updates to the same extent. This can be seen from the fact that initial large deformations are approximated well, but the later steps are too large (Figure 5.8). Therefore, we will use models with multiple layers in the following, where each layer theoretically approximates a single step in the iterative scheme.



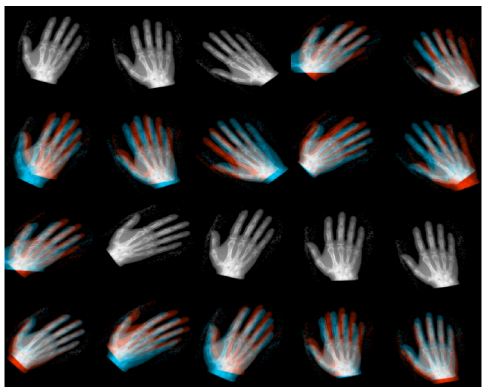


Figure 5.7.: Results for the scheduling experiments. **Left**: $L_1$ distance of the ground truth and predicted grid for the models with no scheduling (None), exponential decay (ExDecay), reduce on plateau (RoP), and cosine annealing (CA). No scheduling works best for a shared weights model. Orange horizontal lines denote medians, boxes show quartiles, and outliers are discarded for better visibility. **Right**: Registration results for selected samples in the test set. The first and third rows display the results for the classical baseline approach and even rows show the results for the model without learning rate scheduling. The model does not yet achieve accuracies on the level of the baseline method.

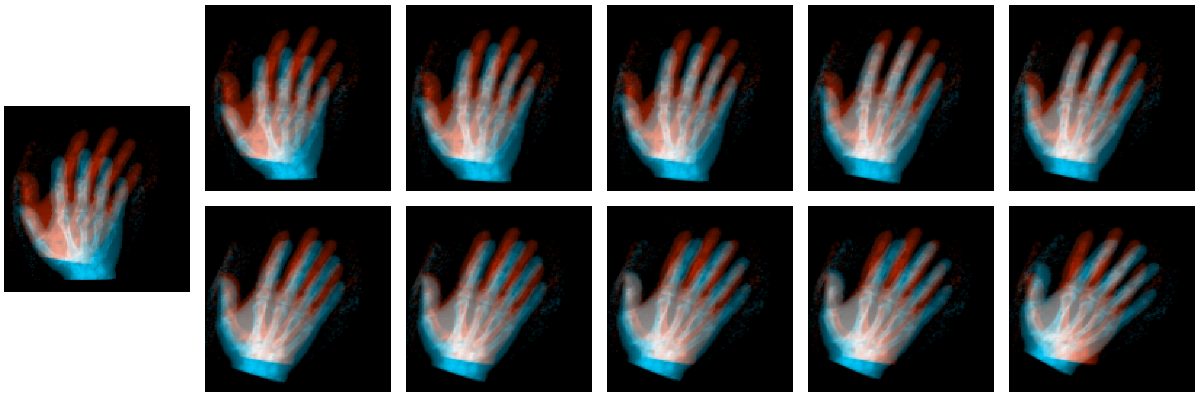


Figure 5.8.: Visualization of the ten steps resulting from the iterative application of the same layer without scheduling. Starting from the initial image pair on the left side, each step is displayed as an overlay of the template (blue) and reference (red) image. White-colored objects mark a good match between the template and reference. The model is able to correct the largest motion but struggles to find the global minimum, applying too large updates between steps five to seven.

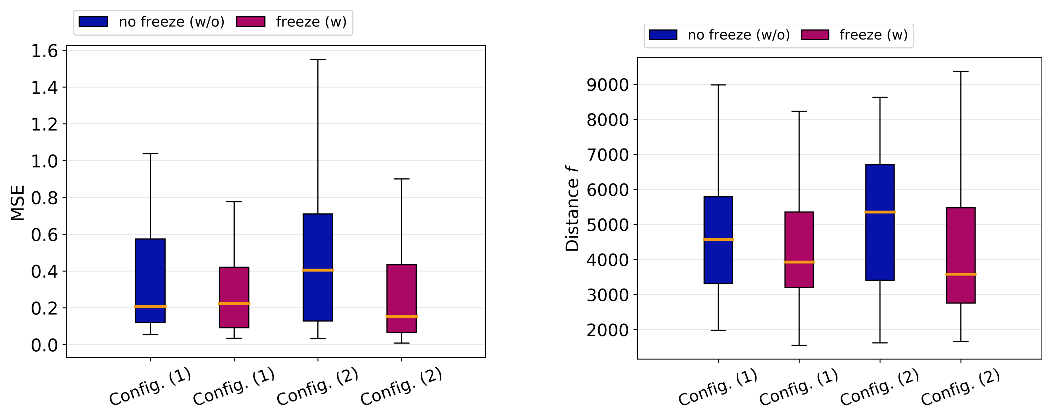


Figure 5.9.: Results for the freezing strategies. **Left**: Mean Squared Error between the ground truth and predicted deformation grid. Two hyperparameter configurations for a 5-layer model with (w) and without (w/o) progressively trained weights are displayed. For both configurations progressively freezing layer weights benefits the accuracy, indicated by the smaller quartiles and smaller minima. **Right**: Regarding the distance function $f$, freezing the weights again results in a better fit with smaller median values, smaller quartiles, and smaller minima. Orange horizontal lines denote medians, boxes show quartiles, and outliers are discarded for better visibility.

**Freezing Strategies.** To retain the iterative structure of classical methods, we evaluate whether progressive training by freezing the layer weights benefits model accuracy. A model with 5 layers is trained progressively, fixing the weights of each layer after 50 epochs are reached. The same model is then trained without this strategy while maintaining the hyperparameter configuration. We provide results for two exemplary hyperparameter configurations:

(1) A 5-layer model trained for 250 epochs with a batch size of 100, a learning rate of $10^{-3}$, and the *Deform Loss* ($\mathcal{L}_D$).

(2) A 5-layer model trained for 250 epochs with a batch size of 50, a learning rate of $10^{-3}$, and the *Deform Loss* ($\mathcal{L}_D$).

The samples during training are created “on the fly” with a consistent seed point for the random number generator. During one epoch, 850 samples are fed to the model. No further techniques such as learning rate scheduling or weight regularization are used. The results in Figure 5.9 indicate that progressively freezing the layer weights improves the error between ground truth and predicted deformation grids as well as the distance function $f$. Therefore, we will continue to progressively train the layer weights, following the suggestions in [HGH19].

**Loss.** Next, we test whether considering multiple steps during loss calculation benefits accuracy. For that reason, we compare multiple model configurations using both the

*Deform Loss* $L_D$ and its multi-step variant *Trajectory Deform Loss* $L_T$. In the following, we show two configurations for small and larger scale models resulting from a random search on the hyperparameter settings. The search space includes the number of epochs [100, 500], batch size [25, 100], and learning rate [$10^{-2}$, $10^{-6}$]. The data we use is the same as in the freezing experiments except we added Gaussian noise to the object area in the image in the range of a third of the variance within the image.

The weighting of each step in the loss calculation is described in Section 5.2. The *Trajectory Deform Loss* is intended to guide the model to the solution encouraging meaningful updates. It can be considered as a form of deep supervision as proposed in [FHH22]. In Figure 5.10, we compare the results of two model configurations with different loss variants:

(1) A 2-layer model trained for 100 epochs with a batch size of 50, a learning rate of $10^{-3}$, and the *Deform Loss* ($\mathcal{L}_D$).

(2) A 4-layer model trained for 300 epochs with a batch size of 50, a learning rate of $10^{-3}$, and the *Trajectory Deform Loss* ($L_T$).

The samples during training are created “on the fly” with a consistent seed point for the random number generator. During one epoch, 800 samples are fed to the model. No further techniques such as learning rate scheduling or weight regularization are used. We consider a small model with only two layers and a larger model with four layers. Comparing the boxplots, it becomes apparent that both loss variations result in similar performances. For small-scale models, the *Deform Loss* is able to reduce the maximal error between the deformation grids from 0.8 to 0.6, while the mean and median error is constant with 0.07 and 0.05. But this can be considered within the statistical margin of error while working with neural networks.

A similar trend can be identified for the large model configurations concerning the mean squared error on the translation grid. It should be mentioned that the overall improved performance of the second configuration models can be attributed to the model size and the number of training epochs. It can be assumed that two iterative updates are too few to sufficiently approximate large affine deformations present in our data. As a result, the *Trajectory Deform Loss* does not improve the accuracy while being more memory demanding. Our model design requires keeping track of the graph of all steps in order to calculate the gradient. For smaller models this is feasible but with increased model size we run into memory problems. For this reason, we decide to make use of the *Deform Loss* enabling larger model and batch sizes while we do not have to accept any loss in accuracy compared to the *Trajectory Deform Loss*.

**Benchmarks.** Based on the findings from the previous experiments, we test the performance of our model in comparison to classical registration methods. The following results were first presented in our contribution [KL22] to the *Medical Imaging with Deep Learning* (MIDL) conference in 2022.

All experiments are performed on a 2x6-core Intel Xeon Gold 6128 CPU @ 3.40GHz with 24 logical cores and 3x GeForce RTX 2080 Ti GPUs with 11 GB of memory each.

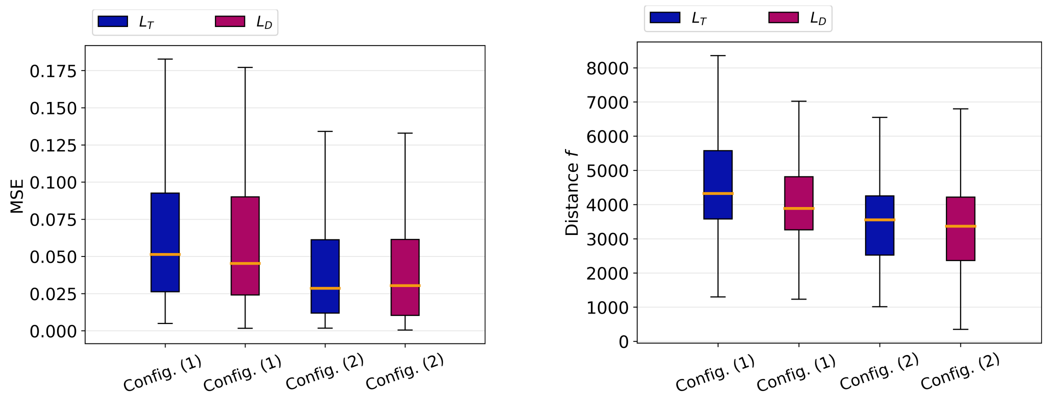


Figure 5.10.: Results for the loss comparison. **Left**: Mean Square Error between the ground truth and predicted deformation grids. On the left, the accuracy on the test set for the two-layer models trained with *Deform Loss* and *Trajectory Deform Loss* is shown. The performance of both loss variations is nearly identical. The right side displays the results for the four-layer models. Again both loss variants display a similar performance. **Right**: The trend is confirmed by the resulting distances $f$. For the small model, the *Deform Loss* displays little improvement but for the large model, there is less difference between both variants with similar median and quartiles. Orange horizontal lines denote medians, boxes show quartiles, and outliers are discarded for better visibility.

We use a simple classical baseline method *(plain)*, where we use a limited-memory BFGS solver with a memory size of $l = 5$ and an Armijo line search for minimizing $f$ directly. The maximum number of BFGS iterations is set to $K = 30$. In our experiments, more iterations did not increase registration quality. Finding suitable initial estimates for the Hessian is a difficult task, we refer to [NW06; Mod03] for a detailed discussion. We follow the proposal in [NW06, Chap.6] and set the initial estimate of the Hessian to the identity matrix scaled by $\frac{1}{\nabla f}$. We embed the solver into a coarse-to-fine approach on 4 levels to obtain minimal robustness against local minimizers. The image resolutions range from $16 \times 16$ to $128 \times 128$.

We also provide a combined *(comb)* method, where we apply the *plain* algorithm to the predictions of our model. The model is trained independently to compute an initial estimate. To further highlight the benefits of predicting initial estimates, we reduce the iterations in our *plain* algorithm to $K = 15$.

Again we employ a random search over the hyperparameters batch size [50, 200], learning rate [$10^{-3}$, $10^{-5}$], and epochs [50, 1000]. We use the Adam optimizer with the default settings for the momentum. Based on our previous results we use the *Deform Loss*, adapt *freezing* to successively train layer weights, and do not utilize any form of *scheduling*. For the affine model, we use a linear projection layer to increase the number of trainable parameters by 4 compared to the input size. We extend the previous measurement of performance by means of MSE with absolute errors of the obtained

Table 5.1.: Mean squared error (MSE) of deformation grids on test sets for *Hands*, *Brain*, and *fastMRI* tasks. By augmenting a simple L-BFGS method *(plain)* with our network *(comb)*, its robustness can be greatly improved, with performance similar to *FAIR* and *elastix* on the specialized tasks (see also Figure 5.12).

| | Hands | | | | Brain | | | | fastMRI | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSE | plain | comb | FAIR | elastix | plain | comb | FAIR | elastix | plain | comb | FAIR | elastix |
| mean | 1.17 | **0.02** | 0.19 | 0.22 | 1.00 | **0.28** | 0.43 | 0.33 | 0.69 | 0.58 | **0.12** | **0.12** |
| median | 0.30 | **0.00** | 0.02 | **0.00** | 0.24 | **0.00** | 0.02 | **0.00** | 0.07 | 0.03 | 0.01 | **0.00** |
| max | 8.28 | **0.23** | 2.76 | 3.53 | 5.43 | **4.69** | 7.69 | 7.13 | 6.62 | **6.33** | 6.60 | 6.80 |

parameters $A$ in the Frobenius norm $\|A - A_{gt}\|_F$ and $b$ in the Euclidean norm $\|b - b_{gt}\|_2$. Both errors are again calculated between the ground truth deformations and the model's predictions.

We compare our model to two benchmark methods: First, the implementation of affine image registration in the freely available *FAIR* toolbox [Mod09]. We use a multi-scale approach with 4 levels similar to our own *plain* method. Here we set no limit on the number of iterations on each level. In addition, *FAIR* provides multiple strategies to increase robustness such as an improved estimate of the initial Hessian, filtering, and regularized interpolation. We report the results for the best-performing parameter setting consisting of an L-BFGS solver, the mean squares (MS) distance measure, and again a coarse-to-fine approach on 4 levels. Second, we compare our model to the *elastix* toolbox. An extensive grid search on the test set to select the optimal distance metric, optimizer, and multi-scale scheme is performed. The search space is described in detail in Section 4.4 of the preceding chapter.

Again we start with rigid deformations using rotations in $[-60°, 60°]$ and translations of at most a quarter of the image dimensions. Image pairs are created following the algorithm described in Section 5.3 using all available data sets (*Hands*, *Brain*, *fastMRI*) and Gaussian noise is added to the reference images ranging up to a third of the variance within images.

The simple baseline method *(plain)* has poor performance regarding the MSE over the samples in the test set (Table 5.1). It displays a much higher mean MSE as well as the highest maximum error. Since the mean can easily be influenced by a small number of large outliers, the poor performance can be attributed to the insufficient robustness of the method. The outliers are caused by suboptimal local minima and failed line searches during optimization. A few representative examples of such outliers are presented in Figure 5.11. Our model alone has also difficulties to achieve accurate registration.

However, our combined strategy (*comb*) yields accuracies comparable to the highly tuned *FAIR* and *elastix* results. What is clearly evident from the fact that the *comb* method achieves the same median MSE as *elastix* on both the *Hands* and *Brain* data sets, even outperforming *FAIR* by a narrow margin of 0.02. Overall *FAIR* and *elastix* show

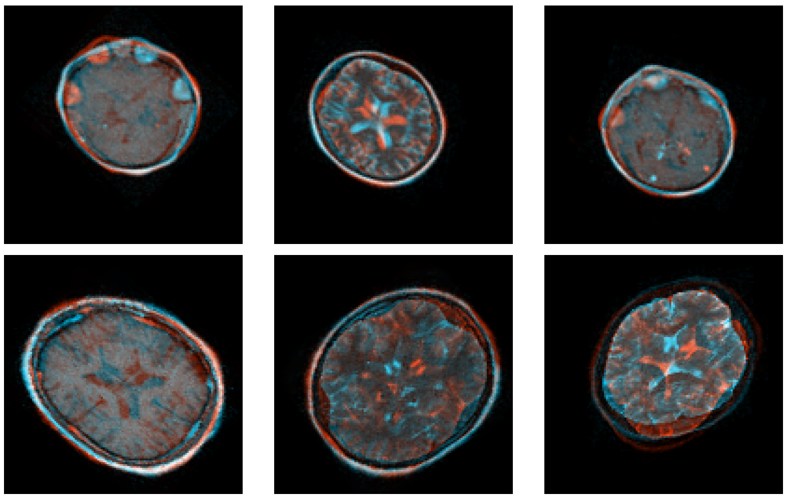


Figure 5.11.: *Elastix* registration results with the worst MSE for the data sets *fastMRI* (**top**) and *brain* (**bottom**). Outliers can be mostly attributed to a high level of symmetry leading to an incorrect initial estimate for the rotation.

good results, but also fail in a significant number of cases, indicated by the relatively high mean values compared to the *comb* approach. This trend is confirmed by the outliers (+) depicted in Figure 5.12. Applied to smaller tasks, our *comb* method learns to avoid extreme outliers and in the case of the *Hands* data set eliminates the outliers entirely. The performance of the *comb* method is less satisfactory when we consider larger tasks such as the *fastMRI* data set. Here the classical methods achieve a lower mean error of 0.12 compared to the 0.58 of the combined approach. We attribute this to a much higher inter-object variance, which poses a more challenging problem for learning-based approaches. The classical methods operate on an instance basis and therefore are less susceptible to increased variance in the data.

A closer look at the difference in accuracy regarding the rotation matrix $A$ and the translation vector $b$ (Figure 5.13, 5.14) reveals that our combined method struggles to predict the correct rotation but is better at predicting translations. For the small data sets *Hands* and *Brain*, it displays higher errors between the ground truth and predicted rotation matrix compared to *FAIR*. On the other hand, this relation reverses for the translation, where the *comb* method outperforms *FAIR* by a large margin (Figure 5.13). This trend holds true in all tasks (Figure 5.14). Again one can see that the *comb* method reduces outliers compared to the two benchmark methods, but overall produces larger errors in the rotation matrix $A$. Results for the translation vector reveal that *comb* is able to outperform *FAIR* on smaller data sets, but lacks the accuracy of *elastix*. We consider the task of learning exact rotations to be particularly challenging in the given setting. The nearly symmetric structure of the brain results in a complex distance function $f$ with many local minima.

So far, we have only looked at rigid deformations. An open question is how our model behaves when additional shear effects occur. We extend our deformations by a shear

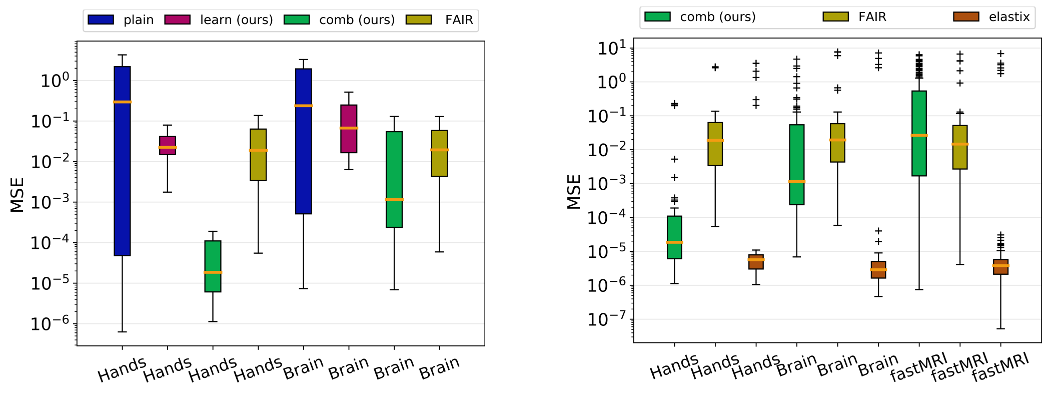


Figure 5.12.: **Left:** When using only either the trained network *(learn)* or the simple baseline method *(plain)* the performance is not competitive. However, when combined *(comb)*, the MSE is comparable to or even better than *FAIR* (see orange median bars; outliers are not shown for clarity). **Right:** MSE of the predicted deformation fields for all tasks. On the smaller *Brain* and in particular *Hands* data sets, our approach allows reducing the number of extreme outliers and yields performance comparable to the established methods (see also 5.1). Orange horizontal lines denote medians, boxes show quartiles, and outliers are marked by + symbols.

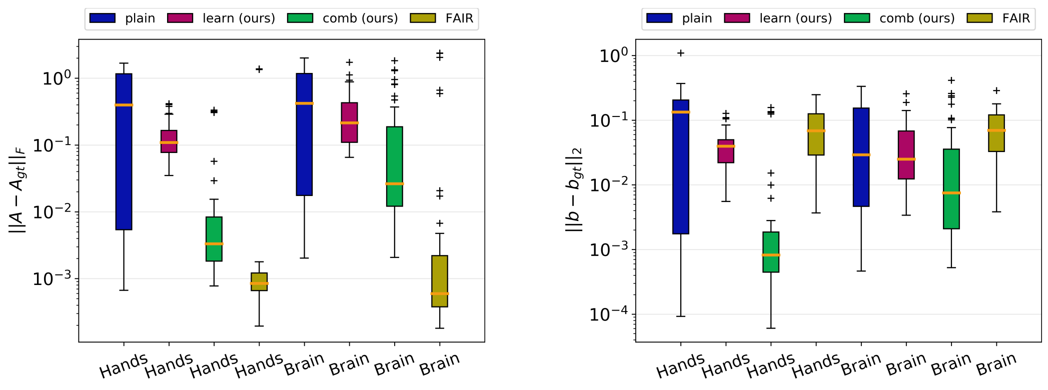
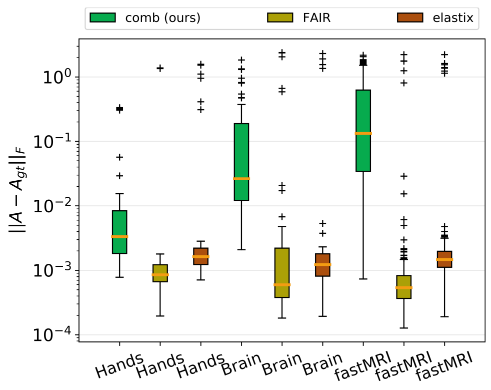


Figure 5.13.: Distance to ground truth of matrix $A$ (**left**) and translation vector $b$ (**right**) for the experiment in Figure 5.12–left. The *learn* method has problems estimating the rotation accurately. Compared to *plain*, it is on average more accurate with less scatter. However, in some cases, the *plain* method achieves a better prediction by up to an order of magnitude. The *comb* approach improves the prediction of rotation but does not reach the level of *FAIR*. Looking at translation, the ratio turns and FAIR performs worse than our model (*learn*). The gap with *FAIR* increases when the *comb* approach is considered. The trend can be seen for both the *Hands* and *Brain* data sets.



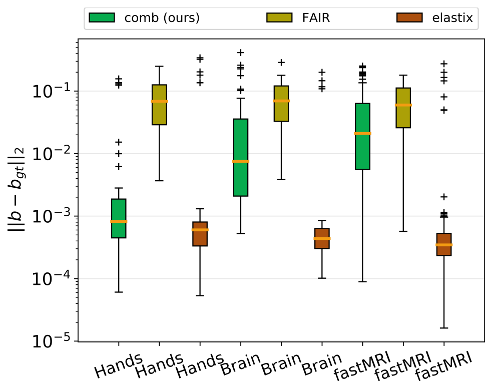


Figure 5.14.: Distance to ground truth of matrix $A$ (**left**) and translation vector $b$ (**right**) for the experiment in Figure 5.12–right. Looking at the prediction of the rotation matrix on the right side, it is clear that the *comb* method has problems determining rotations accurately. For all three data sets, the performance is worse than the two benchmark methods of *FAIR* and *elastix*. One possible reason is that the *plain* method has weaknesses in determining rotation (see Figure 5.13). For translation, the *comb* method achieves higher accuracies than *FAIR* for all three data sets. However, it is beaten by *elastix*, but only by a small margin for the *Hands* data set.

factor up to 15% of the rotation angle. The generation of the new fully affine deformations follows the description in Section 5.3. We repeat the experiments from before with the same hyperparameter settings. The results are listed in Table 5.2. Considering full affine deformations including shearing or scaling shows no loss in performance. For the small data sets, the accuracy even improves slightly, especially for the *Brain* data with reduced mean error and maximum error of 0.04 and 0.56. In the case of *fastMRI*, the mean error increases, resulting from an extreme outlier with an error of 94.30.

**Multi-level Model.** Finally, we investigate the influence of a multi-level (coarse-to-fine) network structure, which is proven to benefit model accuracy in a medical image registration setting [HGH19]. We follow the implementation used in *FAIR* without image smoothing. The ML variant of our model is depicted in Figure 5.3. The multi-level structure improves the network performance. This becomes especially clear for the small data sets (Table 5.3). The multi-resolution model alone achieves near-perfect results on the synthetic *Hands* data set. The greatest improvement is shown for the *fastMRI* data set, where the *comb* model yields the lowest mean and median error of 0.04 and 0.0, outperforming the established benchmark toolboxes. We want to particularly highlight the improvement in mean error compared to *FAIR* and *elastix*. This again underlines that our learned model can reduce the number of outliers.

Table 5.2.: Mean squared error (MSE) of deformation grids on test sets for *Hands*, *Brain*, and *fastMRI* tasks. Shown are the results for full affine deformations: Transformations are created by adding a scalar to each rotation matrix entry randomly sampled from a uniform distribution (15% of rotation angle). The results confirm the findings in Table 5.1 for the purely rigid deformations. The *fastMRI* data set exhibits an extreme outlier, but still, the combined (*comb*) method exceeds baseline (*plain*) performance.

| | Hands | | | Brain | | | fastMRI | | |
|---|---|---|---|---|---|---|---|---|---|
| MSE | plain | learn | comb | plain | learn | comb | plain | learn | comb |
| mean | 1.17 | 0.02 | **0.01** | 1.02 | 0.13 | **0.04** | 0.96 | **0.72** | 1.19 |
| median | 0.31 | 0.01 | **0.00** | 0.39 | 0.04 | **0.00** | 0.72 | 0.29 | **0.02** |
| max | 6.55 | 0.13 | **0.23** | 4.51 | 1.54 | **0.56** | **3.02** | 4.97 | 94.30 |

Table 5.3.: Mean squared error (MSE) of deformation grids on test sets for *Hands*, *Brain*, and *fastMRI* tasks. Shown are the results for a *multi-level (coarse-to-fine)* model architecture. The multi-level structure further improves the results and for small sets such as *Hands* enables our approach to generate near-perfect results. The *learn* procedure produces an extreme outlier in the fastMRI data set, which is even more pronounced in the *comb* procedure. Despite this outlier, the mean of both methods is lower than that of the *plain* method, suggesting that this is an isolated phenomenon.

| | Hands | | | Brain | | | fastMRI | | |
|---|---|---|---|---|---|---|---|---|---|
| MSE | plain | learn | comb | plain | learn | comb | plain | learn | comb |
| mean | 1.17 | **0.00** | **0.00** | 0.98 | 0.23 | **0.21** | 0.69 | 0.49 | **0.44** |
| median | 0.30 | **0.00** | **0.00** | 0.45 | 0.01 | **0.00** | 0.07 | 0.07 | **0.00** |
| max | 8.28 | 0.03 | **0.00** | **3.19** | 4.18 | 6.43 | **6.62** | 7.85 | 8.97 |

### 5.4.2. Summary and Discussion

We introduced a new architecture to incorporate the iterative nature of classical optimization into neural networks. Based on the structure of the model, we examined how different scheduling techniques affect performance. It was found that these techniques had little effect on the accuracy of the registration. To best approximate the iterative nature of classical methods, we applied successive training of the weights. This significantly improved the accuracy of our approach.

Finally, we evaluated our model's capability to predict robust initial estimates for simple non-tuned optimizers *(plain)*. We measured the performance of the *combined* method in comparison with two established frameworks *FAIR* and *elastix*. The results demonstrated that our model can predict robust starting points for the *plain* optimizer. We focused extensively on large deformations up to 60 degrees of rotation and translations in the range of a quarter of image dimensions, which are especially challenging. If the distribution of possible deformations can be derived from the data, our supervised approach enables good registration results in a few steps without expensive parameter tuning.

We proved that combining both classical and learning-based methods is beneficial, as is evident from the reduction of outliers. This trend is reflected by current developments in the field of image registration [FHH22; Jia+21; Xu+21; Qiu+22]. In the classical setting, outliers occur mostly due to a failed optimization, where no descent direction is found. An exciting connecting task is the transfer of this promising strategy to non-linear deformations, a more frequent problem in medical imaging. Currently, we also have not considered more specialized data terms or auxiliary loss functions, which are frequently used [HGH19; FHH22; Her+19].

## 5.5. Non-parametric Image Registration

The first problem that arises when moving from affine to *non-parametric deformations* is the increase in memory requirements. For two-dimensional images of size $128 \times 128$, $x_k$ grows to a size of $32\,768 \times 1$, if each grid point is to be individually and locally transformable. In combination with the need to collect several samples in one batch to train effectively, the choice of the *Fuse Layer* and *Scale Layer* are of high importance. In addition, we need to add a regularizer to our objective function $f$ in order to avoid foldings and thus ensure the regularity of the grid. Gradients have to be determined with regard to the regularizer and passed to the network. In our model, this results in a doubling of the input channels. This is because the gradients of each image direction are represented by a separate channel. In the context of this thesis, we have chosen the curvature regularizer, as it is a pragmatic and successful choice in practice [FM03].

### 5.5.1. Experiments and Results

As the size of our model is the most pressing problem in the non-parametric setting, we first look for a suitable solution to reduce the size of the iterated $x_k$ and subsequently

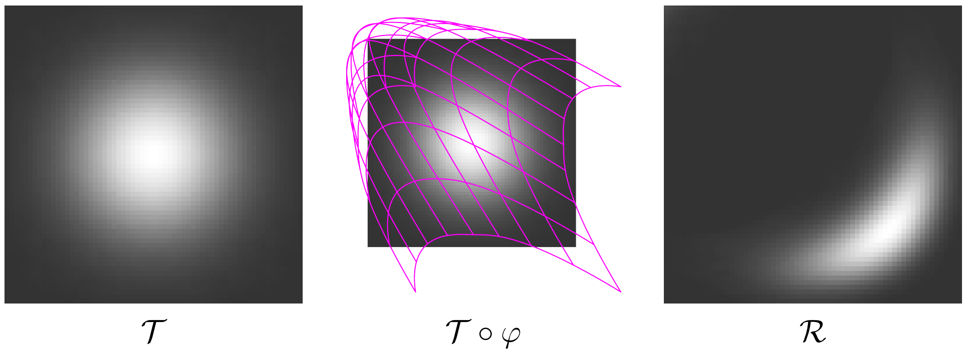


Figure 5.15.: Example image pair from the *Blobs* data set. **Left**: The template image $\mathcal{T}$ is a two-dimensional Gaussian distribution. **Center**: The transformation $\varphi$ (grid) is used to deform the template image. We intentionally created very strong non-parametric deformations to challenge our model. **Right**: The result is the reference image $\mathcal{R}$.

of the model itself. For this purpose, we again construct a synthetic problem with very large non-parametric deformations and simply structured objects in the form of two-dimensional Gaussian-shaped ellipses. For the sake of simplicity, we will call these *Blobs* in the further course of the thesis. An example of such a blob object is provided in Figure 5.15.

**Projection Layer.** Using this data, we search for a suitable design for the two *projection layers*. For the *Fuse Layer*, we use a two-dimensional convolutional layer with a kernel size of $k = 5$. We initialize the weights with $\frac{1}{k}$ so that all elements of the input contribute equally. The weights of the layer are not trainable, in all experiments with trainable weights the loss did not reasonably converge. The number of input channels depends on the objective function. For the configuration with an SSD data term and a curvature regularizer, it follows that the inputs have two channels, which are reduced to a single-channel output. A number of different values for the regularization weight $\gamma$ have been tested and the best results have been obtained with $10^{-5}$. This value has been adopted for the following experiments. The weight enters the model calculations only via the gradients of the inner distance function $f$, it is not explicitly defined as a trainable parameter. Finally, the output of the convolution layer is converted to a one-dimensional vector to fit the LSTM requirements.

To generate a dense displacement field from the one-dimensional output of the main block, we use another layer. The *Scale Layer* works in the opposite way to the *Fuse Layer*, generating a deformation grid from the outputs. It is especially challenging to guarantee the regularity of the resulting grid. We are therefore testing four different variants for the *Scale Layer*: an up-convolution layer, a Gaussian smoothing layer with trainable mean and variance, bilinear interpolation followed by a trainable smoothing layer, and bilinear interpolation.

To test the different variants of the scale layer, we exchange the corresponding layer

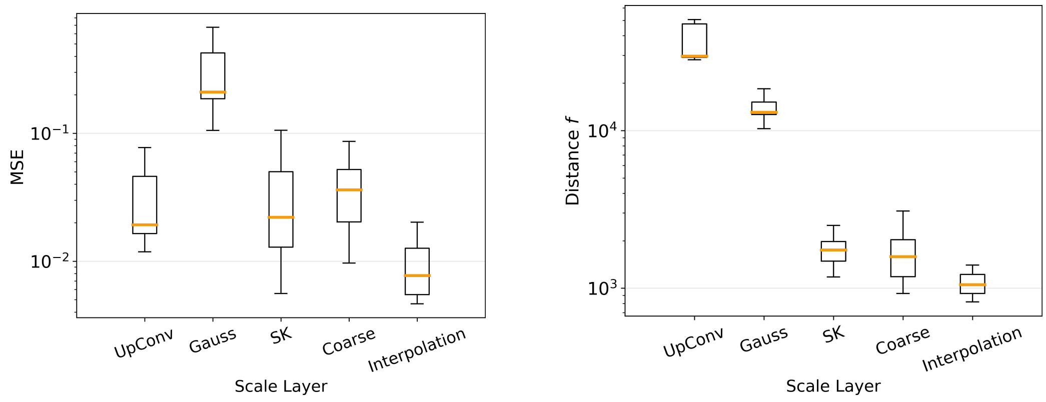


Figure 5.16.: Results on the *Brain* data set for the different versions of our Scale Layer: Up-convolution (UpConv), Gaussian (Gauss), trainable smoothing layer (SK), and two forms of interpolation on full-scale (Interpolation) and coarse inputs downscaled by the factor 4 (Coarse). **Left:** Mean squared error of deformation grids on the test set for different forms of the Scale Layer. The layers based on interpolation produce the lowest errors. Of the trainable layers, the UpConv layer achieves comparable error rates to interpolation. **Right:** The value of the distance function $f$ averaged over all samples in the test set for the five tested layer types. The quality of the registration is best when interpolation layers are used. Up-convolution creates blocks of pixels the size of the kernel, producing a checkerboard pattern in the image, resulting in large distances. An example image can be found in Figure 5.17.

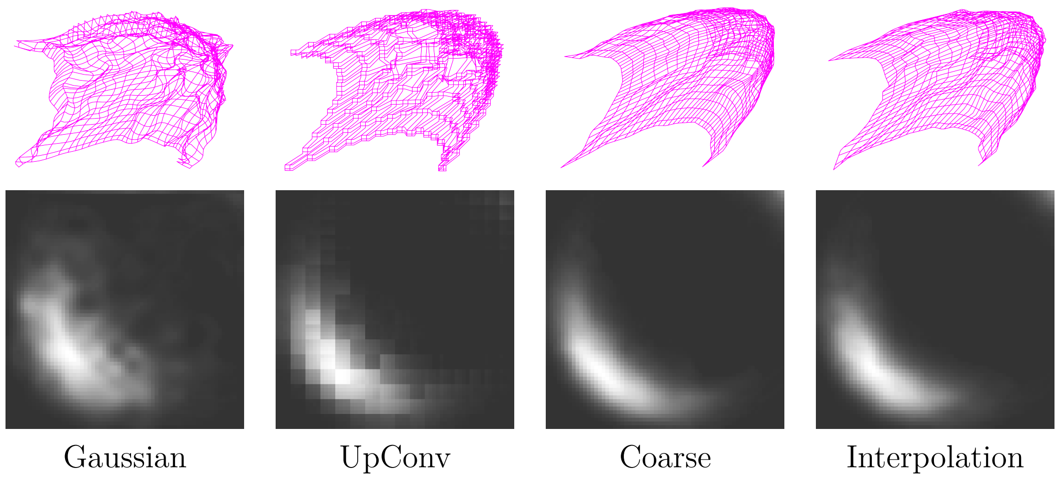


Figure 5.17.: Exemplary deformation grid for different layer variants (**upper row**). The interpolation-based variants produce the smoothest grids and the best image quality (**lower row**). The Gaussian layer results in many folds within the grid, which in turn leads to blurred checkerboard patterns in the image (**far left**). Strong checkerboard patterns are also the result of the UpConv layer and step patterns are also visible in the generated grid (**center left**). These are the result of the layer design, as the UpConv layer maps one input value to multiple outputs.

in each case in a model consisting of three main blocks. We train all models with a learning rate of $5 \cdot 10^{-4}$, a batch size of 250, and use the *Deform Loss*. Due to the applied early-stopping, the number of epochs varies slightly between 120 and 150, which is the maximum number of epochs. Again, the quality of the registration is determined by the error between the predicted grid and the ground truth. In addition, we also consider the value of the distance function $f$, which classically serves as a quality criterion in image registration. The results are shown in Figure 5.16.

Looking at the error in the deformation grids, the variants based on interpolation perform best, and only the trainable UpConv layer achieves a similar level of accuracy. For inputs at full resolution and interpolation as a scale layer, a mean error and median of 0.01 are achieved. The UpConv layer manages to achieve a mean error of 0.04 and a median of 0.02. If we reduce the size of the inputs by a quarter (Coarse) and use interpolation, a mean error of 0.08 and a median of 0.04 is still achieved. A major drawback of the trainable UpConv layer is revealed when we consider the distance function $f$ and the resulting grids (Figure 5.17). Here, the model with the UpConv layer loses accuracy and produces the largest errors with a mean distance of 35 553 and a median of 29 653. In comparison, both the interpolation and coarse variants produce much smaller values with a mean distance of 1 177 and 1 842.

**Grid Resolution.** The use of interpolation allows us to extend our model to non-parametric deformations. The next logical step is the application to *full resolution grids*

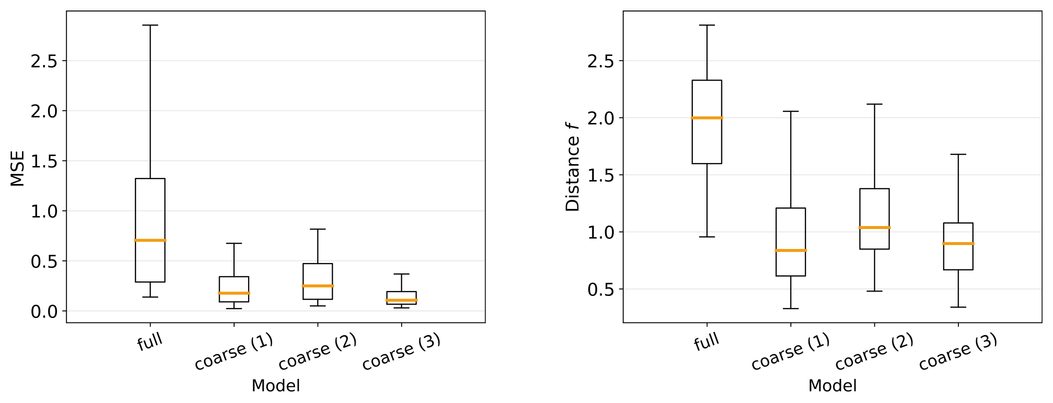


Figure 5.18.: MSE and distance function $f$ for the experiments from Table 5.4. **Left**: The error between the deformation grids is smaller for the models with coarser grids, but these were also trained for twice as long. **Right**: Looking at the distance, a similar picture emerges, again the coarser grids and longer training times produce better results.

and images. We focus on large deformation consisting of an affine and non-parametric part. It quickly turns out that new problems arise from this step. The size of the full resolution grids forces us to use smaller batch sizes and training is slower: a 4-layer model on full resolution grids takes over 18 hours to train 300 epochs. This is not only due to the larger number of weights whose gradient needs to be determined but also because generating new samples for the batches takes more time. However, by using a coarser grid and images, it is possible to train a 6-layer model with twice as many epochs in 14 hours.

To converge quickly and reliably, our model requires many training samples. For smaller models, we can provide these through larger batch sizes. The model applied to grids with full resolution does not provide good solutions yet. Coarse resolution inputs initially lead to better solutions (Figure 5.18) and allow the use of larger models and batch sizes. As a result, we can run more experiments in less time with the goal of improving the model's performance for large deformations in the non-parametric setting. During our experiments, we found that a large number of epochs together with a very large batch size leads to better results, see Table 5.4. It would be interesting to see if good results are also possible at full resolution. Unfortunately, within the time frame of this thesis, a more thorough investigation was not possible. The training times in Table 5.4 show that it takes up to twice as long to train a model on full resolution grids. This is also due to the implementation used to generate the training examples. In the non-parametric case, we use a B-spline fit with randomly selected support points. This can lead to no solution being found and new points having to be selected. In the case of fully resolved grids with a larger number of support points, this case is more likely to occur. In addition, for small batch sizes, more runs are required to reach the specified number of samples per epoch. To identify possible weaknesses in our model, we use the lower resolution inputs as they allow us to test different hyperparameter configurations

Table 5.4.: Experimental settings for the results displayed in Figure 5.18. Given are important parameters concerning the model size (Layers, Grid Size) and memory consumption. Grid Size denotes the size of the input vector, in parenthesis the size of the sequence inside the model is given. For the full model, we reduce the size by 4 using the Fuse Layer. The input size is calculated for a batch size of 55 to enable a fair comparison. MB denotes the memory requirements in megabytes during training.

| Model | Layers | Grid Size | Model Size (MB) | Input Size (MB) | Epochs | Time (train) |
|---|---|---|---|---|---|---|
| full | 4 | 32768 (8192) | 1024.501 | 13.75 | 300 | 18:23:08 |
| coarse (1) | 3 | 2048 (1042) | 48.094 | 0.8593 | 600 | 18:45:10 |
| coarse (2) | 6 | 2048 (1042) | 98.192 | 0.8593 | 600 | 14:33:07 |
| coarse (3) | 6 | 2048 (1042) | 98.192 | 0.8593 | 600 | 12:21:46 |

more quickly.

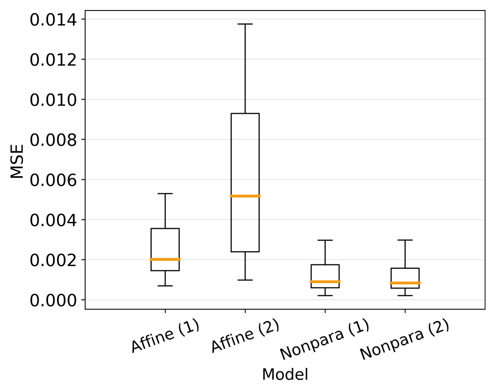
**Large Deformations.** Even smaller grids and longer training times do not lead to successful compensations of large deformations in the non-parametric environment. In order to better determine the reasons for inadequate results, we examine the global and local parts of the transformation separately. We consider the ability of the model to solve three tasks: the compensation of global deformations (Affine), local deformations (Nonpara), and finally the combination of both components into large non-parametric deformations (Large). First, we use our non-parametric model to solve the tasks Affine and Nonpara (Table 5.5). Our goal is to be able to compensate for large non-parametric deformations with a single model. Therefore, we start with our non-parametric model, which operates on the deformation grids.

Figure 5.19 shows the resulting mean square errors between the predicted grids and the ground truth. For the two tasks Affine and Nonpara, our model is able to provide good solutions. Visual inspection of some examples in the test data set also confirms this result (Figure 5.20). Unfortunately, large deformations (Large) cannot be compensated with the same accuracy. For both parameter configurations, the mean error and median are 0.03. Thus, although the full-resolution problem (Figure 5.20) cannot currently be solved satisfactorily, we conclude from the low-resolution problem that it is worth investigating if a combination of affine and non-parametric estimates could solve the global non-parametric problem. A possible solution could consist of combining our affine model (Section 5.4) and the non-parametric model.

We conclude by comparing our model's performance on Nonpara and Large with the two benchmark methods. In *FAIR* we use the provided multi-scale approach with two levels adapted to the low resolution images in both tasks. The default implementation includes an affine pre-registration of the images alongside the strategies to increase robustness such as filtering or regularized interpolation. We again chose the L-BFGS optimizer and set the regularizer weight $\gamma$ to $10^{-5}$. In the case of *elastix*, the grid search resulted in the best-performing parameter setting: adaptive stochastic gradient descent solver, the mean

Table 5.5.: Parameter settings and training times for the non-parametric image registration. We investigate large deformations in a threefold manner: Our model is applied to purely affine deformations with no local components (Affine), non-parametric local deformations with no affine part (Nonpara), and large deformations consisting of both an affine and non-parametric component (Large). For each scenario, we list the two different parameter settings that achieve the best performance on the test set.

| Task | Layers | Batch Size | Learning Rate | Epochs | Time (train) |
|---|---|---|---|---|---|
| Affine (1) | 6 | 2500 | $10^{-3}$ | 2000 | 20:28:26 |
| Affine (2) | 6 | 2500 | $5 \cdot 10^{-4}$ | 2000 | 21:21:40 |
| Nonpara (1) | 6 | 2500 | $10^{-3}$ | 1000 | 65:52:09 |
| Nonpara (2) | 6 | 1250 | $10^{-3}$ | 1000 | 66:10:02 |
| Large (1) | 6 | 2500 | $10^{-3}$ | 1000 | 65:42:19 |
| Large (2) | 6 | 2000 | $10^{-3}$ | 2000 | 66:17:45 |



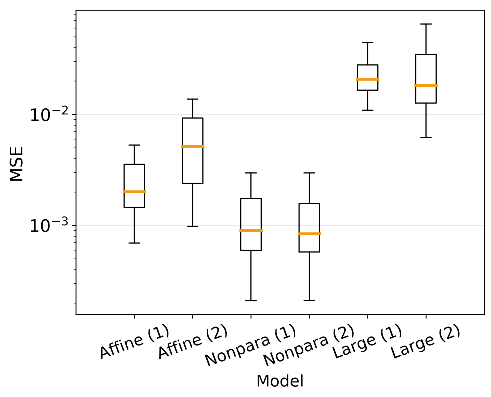
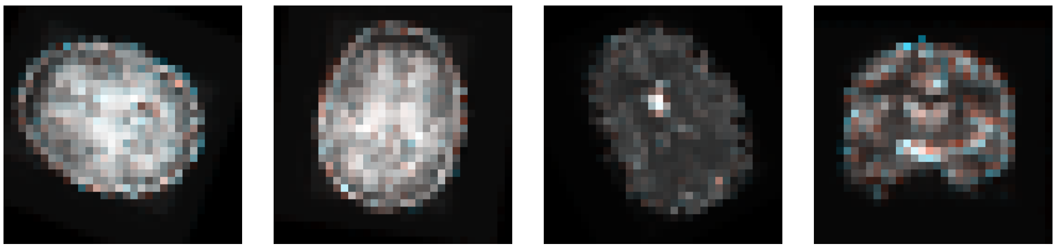
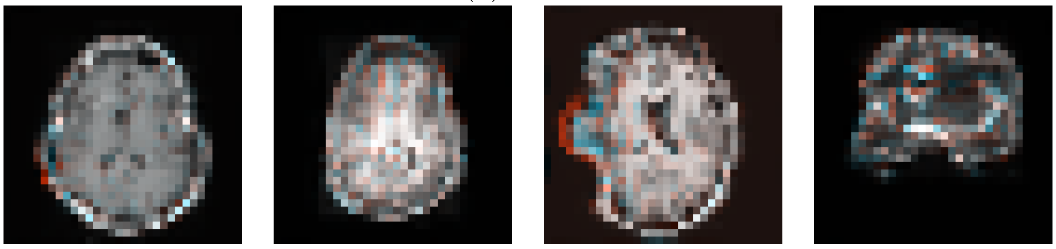


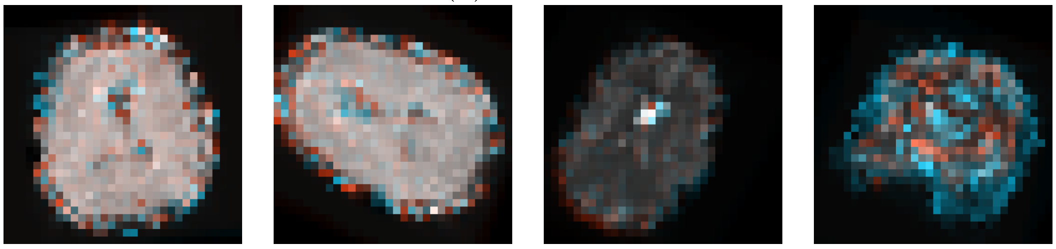
Figure 5.19.: Results for our experiments in Table 5.5. Displayed is the MSE on the deformation grids. **Right**: Errors for the single components *Affine* and *Nonpara*. Both tasks can be solved sufficiently if we separate them and apply the model to each problem individually. **Left**: Given the same amount of time it took to complete the individual tasks, the same model struggles to complete the combined task with a similar level of accuracy. Given the complexity of the deformations, the results that our model has already achieved are a promising start to pursue the approach further in the future.



(a) Affine



(b) Nonpara



(c) Large

Figure 5.20.: Overlay of the transformed template image (blue) with the reference image (red) for randomly chosen samples of the *Brain* data set. At points of high correspondence, the two add up to a greyscale image. (**a**) For the task of affine deformations, the non-parametric model achieves good results, as can be seen from the few blue or red spots in the overlayed images. (**b**) A similar picture emerges for transformations with only local (non-parametric) changes. However, it can be seen that very large changes, as in the case of the bottom left, cause difficulties. (**c**) When combining both types of transformation — i.e., creating large deformations with global and local components — the problem becomes much more difficult. Although an optimal solution has not yet been found, our model shows great potential for large deformations.

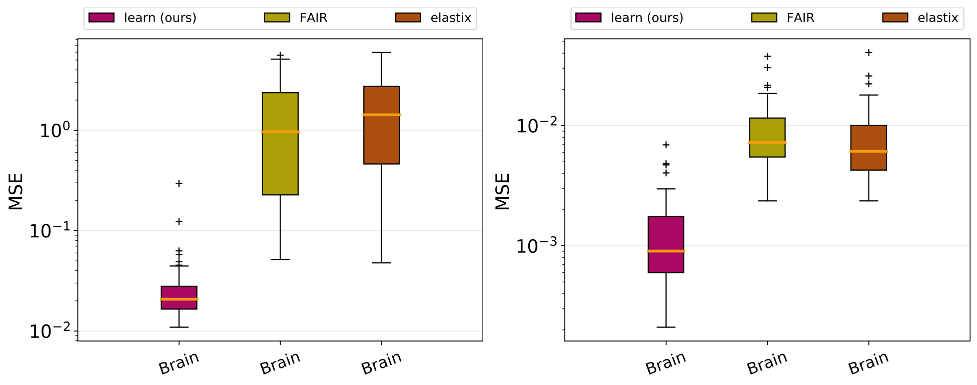


Figure 5.21.: MSE between the deformation grids for our best model (learn) and the two benchmark toolboxes (*FAIR*, *elastix*) on the Large task (**left**) and the Nonpara task with only local deformations (**right**). Our method displays considerably better performance than both *FAIR* and *elastix*. We constitute that to the fact, that the combination of global and strong local deformations hinders a successful affine pre-registration, suggested by the failed registrations displayed in Figure 5.22.

squares (MS) distance measure, and again a coarse-to-fine approach on two levels. We extended the method to non-parametric deformations by the successive execution of an affine registration and the default B-spline based non-parametric deformation.

Both approaches have major problems with the Large data set. Particularly noticeable are the high errors between the ground truth and predicted grid for both *FAIR* and *elastix*. The reason for this is probably a failure of the pre-registration as suggested by the examples with the largest errors (Figure 5.22).

The comparison with the benchmark methods once again emphasizes that the task of dealing with large deformations is a very demanding one. It should be noted that both classical methods are not designed to be used without considerable fine-tuning of the parameter settings. Especially for the complex deformations in the Large task, an intensive examination of individual examples is recommended. In particular, the regularization weight $\gamma$ should be checked individually for each image pair, given the large variability in the strength of local deformations. The application of classical variational methods to reduce such variable deformations requires a high degree of expertise on the part of the user. Since this elaborate process also requires a deep understanding of both toolboxes and our trained model already produces superior results, we propose to use the outputs of the network as initial estimates. In this way, the complexity of the deformations is reduced and the classical methods do not have to compensate for the affine parts anymore. If the distribution of transformations is known or can be inferred from examples with minimum and maximum deformations, using our network as a pre-registration can probably support classical methods.

We consider the results in Figure 5.21 to be an indicator that learning-based methods with their nonlinear updates can support classical image registration approaches with an unsolved problem: the dependence of the methods on robust initial estimates.

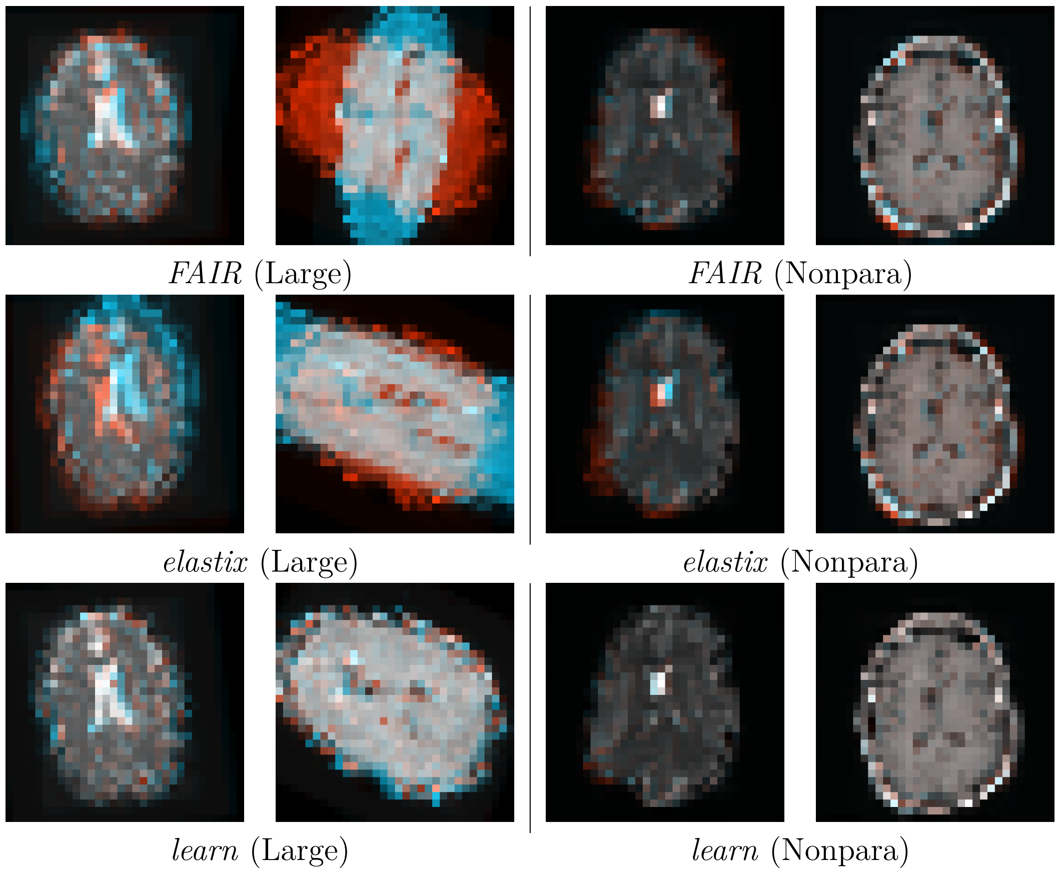


Figure 5.22.: Example results of all three methods for the two tasks Nonpara and Large. In the case of large non-parametric deformations with affine components, the two classical approaches give inadequate results. On the left, a good result (**far left**) and the result with the largest error between the grids (**center left**) are shown as examples. It is clear that the methods do not produce satisfactory solutions when the affine parts are not well balanced. The small regularization weight allows strong decomposition effects but is necessary to allow strong local deformations (**center right**).

### 5.5.2. Summary and Discussion

Having successfully applied our model to affine deformations, we looked at extending the methodology to the non-parametric case. The design of the *Scale Layer* was particularly important. The size of the inputs made it necessary to reduce them considerably using the *Fuse Layer*. To reconstruct regular, dense grids from the model outputs, various methods were tested, from linear interpolation to trainable convolution kernels. To do this, we created large non-parametric deformations and applied them to simply structured 2D Gaussian distributions. Two conclusions can be drawn: First, it is possible to compensate for strong non-parametric deformations with our model. Second, these problems can be solved most efficiently by using coarser resolution grids and images in combination with linear downscaling and upscaling to keep the number of parameters in the model small.

We extended our model with a layer that linearly interpolates the outputs to create dense grids. We also decided to work with coarser resolutions as this allowed more experiments to be run in less time. We then investigated the ability of our model to solve large deformations consisting of a global affine and a local non-parametric part. Such deformation could not be satisfactorily compensated for by the model. However, it was shown that, with few exceptions, the individual components separately produced registrations of high accuracy. Considering that trying to solve such large deformations with a single model is an ambitious goal, this is a respectable achievement. These results offer a promising perspective, which we leave for future work.

## 5.6. Conclusion

The second part of this thesis dealt with the application of neural networks in image registration. The focus was on **merging two different approaches**: the classical variational methods in the form of **iterative update schemes** and **neural networks**, which usually register two images in only one step. Our aim was to combine the advantages of both approaches by embedding the iterative update scheme in a neural network: accuracy and speed.

The development of a suitable model has been the starting point in Section 5.2. At the heart of our model is an iterative update scheme that follows the methodology of classical methods. The update is no longer the gradient of the distance function multiplied by a variable step size but calculated by a neural network using the gradient as input. The network consists of projection layers and LSTM layers, which we chose for their hidden state functionality. These allow information from previous steps to be included in the computation of the update. To compute the gradients for the internal optimization problem, we use the `autograd` functionality of PyTorch. This means that in addition to our model, the inputs must also be part of the graph, which increases the memory requirements of our method.

Since we aimed to elevate the performance of classical iterative methods using a learning-based method, we decided to adopt a supervised learning approach. Ground truth is hard to come by in an image registration setting if it exists at all. We avoided this problem by creating our own ground truth by adopting the idea for ground truth creation proposed in [Dos+15]. In Section 5.3, we provided two methods to randomly generate feasible affine and non-parametric deformations.

Using the artificial ground truths, we were able to train the model using a supervised approach. Initially, we focused on **affine deformations** because they have a global effect, are controlled by a few parameters, and represent a simpler problem (Section 5.4). To compensate for large deformations, not only the model but also the optimization strategy for adjusting the weights was crucial. First, we experimented with different forms of scheduling. It turned out that common scheduling methods did not have a positive effect on the training, so we discarded scheduling as a whole. In a second step, we established stepwise training of the individual layers, as this method proved to be advantageous [HGH19] in image registration. The successive training of each layer allowed the model to register more accurately with the same parameter configuration, as shown by the smaller errors between the grids and the smaller SSD distance between the images.

Deep supervision is used in multi-layer networks to incorporate information from different resolutions or levels into the loss calculation. Based on this, we tested a loss variant that used all steps of the iterative scheme for the calculation. However, the *Trajectory Loss* did not improve accuracy and was discarded. Our final model was composed of the architecture in Section 5.2, successive training of each layer, and a quadratic mean error on the deformation fields loss function (*Deform Loss*).

Finally, we tested our model on one synthetic and two real data sets and compared the results with two established toolboxes for classical registration methods: *FAIR* and *elastix*. Our model (*learn*) approximated 5 steps of an iterative procedure and then the predictions were passed as initial estimates to a simple L-BFGS procedure (*plain*). For two data sets, this combined approach (*comb*) was able to match the performance of the established methods while producing fewer outliers. Furthermore, a multi-level variant of our model was able to achieve almost perfect results on the synthetic data, outperforming the most accurate method in the *elastix* toolbox.

As an outlook, the interesting class of **non-parametric deformations** was examined in the last Section 5.5 of this chapter. First, we focused on redesigning our projection layers, as fully-resolved grids already require a lot of memory in the two-dimensional case. The best option proved to be linear interpolation to control the size of the inputs and outputs. In this way, we were able to keep the number of weights from becoming too large, while still allowing the prediction of dense regular grids.

When applied to a synthetic example in the form of two-dimensional Gaussian distributions, our model was able to successfully compensate for strong non-parametric deformation. In a further step, we tested the capabilities of the model on large deformations consisting of an affine and a non-parametric part. Some limitations had to be accepted. The input resolution was reduced from $128 \times 128$ to $32 \times 32$ to allow for large batch sizes and long training times. The original deformations are not fully compensated by our network. Looking at the individual components, the model was quite capable of producing accurate registration results.

The approach taken in this thesis, combining classical and learning-based methods, has produced promising results. To realize its full potential, further work can address the conceptual weaknesses identified.

Predicting dense displacement fields using the outputs from the LSTM layer is difficult. Instead, one could adapt approaches like in [San+19] and use the outputs as parameters of a set of basis functions that generate the deformation grids.

The LSTM layer was not designed for image data. A change to convolutional LSTM [Shi+15] could help to better meet the requirements of image registration and exploit the native spatial dimensions of images. Alternatively, the current model can be adapted in the style of [Dos+21] by passing the grids as a sequence of patches.

The implementation in double precision format costs a lot of memory. It ensures a smooth transition between the learning procedure and the classical optimization (Section 5.4), but it is not mandatory in the imprecise learning procedure. In order to reduce the memory requirements of the models, it can be investigated whether the single precision format leads to a loss of accuracy.

Predicting large deformations with a single non-parametric model was very ambitious. A combined model consisting of a subnetwork to compensate the affine parts and another subnetwork to predict the local deformations is a possible solution.

# 6. Discussion

In this thesis, we investigated different applications of neural networks in the fields of medical data analysis and image registration. In the first part, we developed new models to analyze mass-spectrometry imaging data. This includes several pioneering works [Kle+19; Kan+23] for the classification of different tumor types, which differ significantly from the current mostly linear methods (Chapter 3).

**MALDI-MSI.** Mass spectrometry imaging, especially MALDI-MSI, is becoming an indispensable tool in personalized medicine. In contrast to the acquisition method itself less attention has been paid to data processing and the literature is dominated by classical machine learning algorithms. This motivated us to apply neural networks to this data, as they have been very successful in many areas of computer vision. The models presented in Chapter 2 were used to develop new methods in the field of mass-spectrometry imaging. Our methods allow precise classification of tumor subtypes (Chapter 3, Section 3.5) as well as discrimination of different tumor types (Section 3.6, Section 3.7). In doing so, they achieved a higher level of accuracy than established approaches.

- We searched for suitable architectures (Section 3.5). We also investigated several feature extraction methods and were finally able to distinguish ovarian cancer subtypes in a first pilot study. Our method consisted of a flat CNN model applied directly to the unprocessed spectra. The stability of the prediction was increased by eliminating tissue samples with few associated spectra and applying majority voting to obtain predictions for individual patients. In our experiments, CNN achieved the best results compared to classical methods such as SVM.

- In Section 3.6, we extended our methodology to include the two network architectures of Residual Networks and Recurrent Networks. Both achieve good results for sequential data and since MALDI-MSI uses a time of flight detection, a sequential structure of the resulting data can be assumed. In the classification of two types of amyloidosis, the two new architectures performed better than the previously best-performing CNN model. It also showed that the classical methods, such as LDA or SVM, were not able to accurately classify the two types of amyloidosis, even with feature extraction. Furthermore, we discussed that many studies in the field of MALDI-MSI do not cleanly separate their training and test data and in this way falsely overstate the accuracy of their methods.

- In the following Section 3.7, we addressed the problem of noise caused by spectra that carry little or no information. In the case of the pancreatic tumor classification data set, we identified over 2000 such spectra, representing 13% of all spectra. We also investigated the suitability of the Transformer architecture for use in MALDI-MSI. The Transformer model together with a residual model gave the best results. The accuracy of both models exceeded that of traditional univariate methods, demonstrating once again the advantage of multivariate neural networks over the univariate methods widely used in MALDI-MSI.

During this thesis, several challenges were identified and solved. The use of neural networks in MALDI-MSI is an under-researched topic. As a result, there is little work on which to base the methodology. Apart from [Beh+17], we were the first to apply neuronal networks directly to the $m/z$ spectra in MALDI-MSI. The search for suitable models has therefore been a major part of this work and is still ongoing. The spectra in MALDI-MSI are very large 1D data, which in practice are greatly reduced in size in order to work effectively with them. Furthermore, the ratio of feature size to the number of samples in the data sets is unbalanced. As a result, we had to find ways to avoid severe overfitting when developing the models. In addition, noise from various sources is a problem when applying machine learning techniques to MALDI-MSI data. We were able to remove one of these sources by introducing a filter to determine the information content of the spectra.

We have successfully demonstrated that it is possible to apply neuronal networks to the full $m/z$ range of MALDI-MSI spectra without resorting to extensive dimension reduction or feature extraction methods such as PCA or NMF. Despite these successes of our methodology, there are still many opportunities to make the use of neural networks in MALDI-MSI more effective.

**Image Registration.** There are currently two competing approaches to image registration. On the one hand, there are the classical variational approaches, which achieve very high accuracies but are often complex and computationally expensive. Furthermore, the robustness of the optimization is a concern, variational methods depend on good initial estimates. The other is neural networks, which are fast but less accurate and have problems with large deformations consisting of global affine and local non-parametric components. In Chapter 5, a model was designed to combine both approaches by integrating the iterative nature of variational methods into the methodological framework of neural networks. We applied the model to large deformations, which are especially difficult to compensate for.

- The model was introduced in Section 5.2. We replaced the update step of an iterative method with a neural network. We chose the LSTM layer as the architecture because it has a hidden state that can be propagated through several layers. In this way, information about previous steps is included in the calculation of the current update. We also showed the parallels of our methodology with bi-level optimization and meta-learning methods. Furthermore, we explained advanced strategies such as a multi-level model or the successive training of individual layers. Our supervised learning approach depends on ground truth data, which is commonly not available in image registration. Our solution to this problem was to do our own artificial deformations (Section 5.3).

- As a first step, we used the new model to compensate for large affine deformations (Section 5.4). Many experiments were carried out to find a suitable frame structure for our model. We tried different scheduling techniques, but these did not improve the training. More successful was the successive training of each layer. This method, which we call freezing, reduces the mean error by up to 20%. Our form of deep supervision, *Trajectory Deform Loss*, did not improve the results compared to the mean square error between the deformation grids. Finally, we tested the ability of the affine model to produce robust initial values for simply constructed classical methods. Our approach helps a simple L-BFGS method to achieve a similar level of accuracy as the highly optimized classical methods of the *FAIR* and *elastix* toolboxes. It should be emphasized that the initial values of our network helped to significantly reduce the number of outliers. Particularly good results were obtained when a multi-level variant of the network was used.

- In the context of medical imaging, non-parametric deformations are much more common. Therefore, extending our network to the non-parametric case was the topic of Section 5.5. As the central component of our model is the LSTM layer, we first had to look at ways of efficiently reducing the size of the grids in order to reduce the memory requirements. The key step was to project the model outputs back into the deformation space. Care had to be taken to ensure that the predicted grids were dense and regular. The latter was best achieved by a simple interpolation in the last layer. To reduce memory requirements, a coarser resolution of the inputs proved to be the most appropriate. When applied to large deformations with an affine and non-parametric component, the network could only achieve average accuracies. However, when the global and local parts are processed separately, our model achieves promising results even in the non-parametric setting.

First, we had to embed the existing framework of iterative optimization from the *FAIR* toolbox into a neural network. The *FAIR* toolbox uses fixed derivative operators for determining the gradients. As a consequence, the operators would have to be defined in advance for each distance measure and regularizer, at the expense of flexibility. To overcome this problem, we decided to use the `autograd` function of the PyTorch framework. In doing so, we had to accept that the memory requirements of the method would increase.

Another goal was that the model should not simply be a better optimizer to minimize the distance between the images, but that it should find a good fitting registration. Therefore, we decided not to use the distance measure directly, but to use a supervised approach that penalizes the difference in the deformation grids. As a consequence, we had to build a generator that randomly constructs deformations.

For the class of non-parametric deformations, our model could demonstrate its potential in a small proof-of-concept study. For a single model, it is particularly difficult to compensate for large deformations with global and local components. Therefore, a combination of an upstream affine and a downstream non-parametric registration model is often used. We have shown that an approach borrowing from iterative optimization with trainable updates could be a solution. This assumption still needs to be validated in competition with existing methods. Moving towards these hybrid methods is a strong current trend [FHH22; Xu+21; Jia+21; San+19; Qiu+22] of research in this area.

**Future Work.** All of the methods presented in this thesis have much potential for further development. Although we have succeeded in applying neural networks to **MALDI-MSI**, interesting questions remain:

- In addition to classification in the diagnostic setting, MALDI-MSI is also used to identify new potential biomarkers. An extension of our methodology in the form of *Class Activation Maps* (CAM) [Zho+16] could be used by experts to find new markers in the labeled areas of the spectra.

- In Section 3.5, we investigated exploiting spatial information within the patient sample by combining multiple spectra. While ultimately not satisfying, we believe that additional spatial information, for example in the form of an additional $m/z$ image, could be beneficial. A two-part network combining image information and spectral data would be possible.

- The application of the Transformer architecture in Section 3.7 was only possible with simultaneous dimension reduction in the form of pooling operations. This potential loss of information could be counteracted by using *Sparse Transformers* [Chi+19] with a specially designed attention pattern.

We also see considerable remaining potential in our model for combining iterative methods and neural networks in the field of **image registration**:

- The dense nature of the LSTM layer proved to be a good starting point for further development in our experiments. In particular, the lack of scalability in the non-parametric setting proved to be a problem. Alternative network forms such as convolutional LSTMs [Shi+15] or a combination of CNNs and LSTMs [Don+15] are possible solutions, which we discussed briefly in Chapter 2.

- Furthermore, an extension of the loss function via an external regularizer or auxiliary terms to the images of corresponding segmentations [Her+19] could be used.

- In order to be able to assess the potential of the method more precisely, a more intensive evaluation is recommended, especially with regard to fully resolved input data and in contrast to existing established [Bal+19] and related [Qiu+22; San+19] methods. Also interesting is the behavior when using alternative distance functions or mixed training data consisting of different objects from different modalities.

With this work, we hope to help establish neural networks as an important component of MALDI-MSI for diagnostic purposes. We are convinced that the fusion of these disciplines can permanently change the work of molecular pathology in the future. In addition, we have demonstrated the potential of intersectional models from the fields of variational and learning-based image registration, and hope that future studies will continue this direction of research.

# Bibliography

[Ale+10] Theodore Alexandrov et al. "Spatial segmentation of imaging mass spectrometry data with edge-preserving image denoising and clustering." In: *Journal of proteome research* 9.12 (2010), pp. 6535–6546.

[Ami94] Yali Amit. "A nonlinear variational problem for image matching." In: *SIAM Journal on Scientific Computing* 15.1 (1994), pp. 207–224.

[And+13] Yukio Ando et al. "Guideline of transthyretin-related hereditary amyloidosis for clinicians." In: *Orphanet journal of rare diseases* 8.1 (2013), pp. 1–18.

[And+16] Marcin Andrychowicz et al. "Learning to learn by gradient descent by gradient descent." In: *International Conference Advances in Neural Information Processing Systems.* 2016, pp. 3981–3989.

[AÖ17] Jonas Adler and Ozan Öktem. "Solving ill-posed inverse problems using iterative deep neural networks." In: *Inverse Problems* 33.12 (2017), p. 124007.

[ATS+09] Brian B Avants, Nick Tustison, Gang Song, et al. "Advanced normalization tools (ANTS)." In: *Insight j* 2.365 (2009), pp. 1–35.

[AW15] Michaela Aichler and Axel Walch. "MALDI Imaging mass spectrometry: current frontiers and perspectives in pathology research and practice." In: *Laboratory investigation* 95.4 (2015), pp. 422–431.

[Bal+19] Guha Balakrishnan et al. "Voxelmorph: a learning framework for deformable medical image registration." In: *IEEE transactions on medical imaging* 38.8 (2019), pp. 1788–1800.

[BBH20] Max Blendowski, Nassim Bouteldja, and Mattias P Heinrich. "Multimodal 3D medical image registration guided by shape encoder–decoder networks." In: *International journal of computer assisted radiology and surgery* 15.2 (2020), pp. 269–276.

[BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." In: *International Conference on Learning Representations.* 2015.

[Beh+17] Jens Behrmann et al. "Deep learning for tumor classification in imaging mass spectrometry." In: *Bioinformatics* 34.7 (Nov. 2017), pp. 1215–1223.

[Ben10] Merrill D Benson. "LECT2 amyloidosis." In: *Kidney international* 77.9 (2010), pp. 757–759.

[BFS93] Yoshua Bengio, Paolo Frasconi, and Patrice Simard. “The problem of learning long-term dependencies in recurrent networks.” In: *IEEE International Conference on Neural Networks*. 1993, pp. 1183–1188.

[BH19] Max Blendowski and Mattias P Heinrich. “Combining MRF-based deformable registration and deep binary 3D-CNN descriptors for large lung motion estimation in COPD patients.” In: *International journal of computer assisted radiology and surgery* 14.1 (2019), pp. 43–52.

[BHH21] Max Blendowski, Lasse Hansen, and Mattias P Heinrich. “Weakly-supervised learning of multi-modal features for regularised iterative descent in 3D image registration.” In: *Medical image analysis* 67 (2021), p. 101822.

[BMR13] Martin Burger, Jan Modersitzki, and Lars Ruthotto. “A hyperelastic regularization energy for image registration.” In: *SIAM Journal on Scientific Computing* 35.1 (2013), B132–B148.

[BN06] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[Bos+19] Tobias Boskamp et al. “Using the chemical noise background in MALDI mass spectrometry imaging for mass alignment and calibration.” In: *Analytical chemistry* 92.1 (2019), pp. 1301–1308.

[Bra+18] Freddie Bray et al. “Global cancer statistics 2018: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries.” In: *CA: a cancer journal for clinicians* 68.6 (2018), pp. 394–424.

[Bre+19] Kai Brehmer et al. “Variational Registration of Multiple Images with the SVD Based SqN Distance Measure.” In: *International Conference on Scale Space and Variational Methods in Computer Vision*. 2019, pp. 251–262.

[Bro+20] Tom Brown et al. “Language models are few-shot learners.” In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[Can+11] Emmanuel J Candès et al. “Robust principal component analysis?” In: *Journal of the ACM* 58.3 (2011), pp. 1–37.

[Cao+17] Xiaohuan Cao et al. “Deformable image registration based on similarity-steered CNN regression.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2017, pp. 300–308.

[Cas+17] R Casadonte et al. “MALDI IMS and cancer tissue microarrays.” In: *Advances in cancer research* 134 (2017), pp. 173–200.

[CC11] Rita Casadonte and Richard M Caprioli. “Proteomic analysis of formalin-fixed paraffin-embedded tissue by MALDI imaging mass spectrometry.” In: *Nature protocols* 6.11 (2011), pp. 1695–1709.

[CDS01] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. “Atomic decomposition by basis pursuit.” In: *SIAM review* 43.1 (2001), pp. 129–159.

[CFG97] Richard M Caprioli, Terry B Farmer, and Jocelyn Gile. “Molecular imaging of biological samples: localization of peptides and proteins using MALDI-TOF MS.” In: *Analytical chemistry* 69.23 (1997), pp. 4751–4760.

[Cha+06] Pierre Chaurand et al. “New developments in profiling and imaging of proteins from tissue sections by MALDI mass spectrometry.” In: *Journal of proteome research* 5.11 (2006), pp. 2889–2900.

[Chi+19] Rewon Child et al. *Generating long sequences with sparse transformers.* Tech. rep. OpenAI, 2019.

[Cho+15] Jan K Chorowski et al. “Attention-Based Models for Speech Recognition.” In: *Advances in Neural Information Processing Systems 28.* Ed. by C. Cortes et al. Curran Associates, Inc., 2015, pp. 577–585.

[Cor+19] Yovany Cordero Hernandez et al. “Targeted feature extraction in MALDI mass spectrometry imaging to discriminate proteomic profiles of breast and ovarian cancer.” In: *Proteomics–Clinical Applications* 13.1 (2019), p. 1700168.

[CV18] Zhaowei Cai and Nuno Vasconcelos. “Cascade r-cnn: Delving into high quality object detection.” In: *IEEE Conference on Computer Vision and Pattern Recognition.* 2018, pp. 6154–6162.

[DBS10] Sören-Oliver Deininger, Michael Becker, and Detlev Suckau. “Tutorial: multivariate statistical treatment of imaging data for clinical biomarker discovery.” In: *Mass Spectrometry Imaging* (2010), pp. 385–403.

[Déf+22] Alexandre Défossez et al. “A Simple Convergence Proof of Adam and Adagrad.” In: *Transactions on Machine Learning Research* (2022).

[Dei+11] Sören-Oliver Deininger et al. “Normalization in MALDI-TOF imaging datasets of proteins: practical considerations.” In: *Analytical and bioanalytical chemistry* 401.1 (2011), pp. 167–181.

[Del+16] Daniel Delitto et al. “Nicotine Reduces Survival via Augmentation of Paracrine HGF–MET Signaling in the Pancreatic Cancer Microenvironment.” In: *Clinical Cancer Research* 22.7 (2016), pp. 1787–1799.

[Den+09] Jia Deng et al. “Imagenet: A large-scale hierarchical image database.” In: *IEEE Conference on Computer Vision and Pattern Recognition.* 2009, pp. 248–255.

[Des22] Heather Desaire. “How (Not) to Generate a Highly Predictive Biomarker Panel Using Machine Learning.” In: *Journal of Proteome Research* (2022).

[Dev+18] Jacob Devlin et al. *Bert: Pre-training of deep bidirectional transformers for language understanding.* Tech. rep. Google AI Language, 2018.

[DGM98] Paul Dupuis, Ulf Grenander, and Michael I Miller. “Variational problems on flows of diffeomorphisms for image matching.” In: *Quarterly of applied mathematics* (1998), pp. 587–600.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive subgradient methods for online learning and stochastic optimization.” In: *Journal of machine learning research* 12.7 (2011).

[Don+15] Jeffrey Donahue et al. “Long-term recurrent convolutional networks for visual recognition and description.” In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2625–2634.

[Dos+15] Alexey Dosovitskiy et al. “Flownet: Learning optical flow with convolutional networks.” In: *IEEE International Conference on Computer Vision*. 2015, pp. 2758–2766.

[Dos+21] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.” In: *International Conference on Learning Representations*. 2021.

[Doz16] Timothy Dozat. “Incorporating Nesterov Momentum into Adam.” In: *International Conference on Learning Representations. Workshop Track*. 2016.

[DR04] Marc Droske and Martin Rumpf. “A variational approach to nonrigid morphological image registration.” In: *SIAM Journal on Applied Mathematics* 64.2 (2004), pp. 668–687.

[DS96] John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, 1996.

[Du+16] Xiaogang Du et al. “A parallel nonrigid registration algorithm based on B-spline for medical images.” In: *Computational and mathematical methods in medicine* 2016 (2016).

[EN08] Ellen C Ebert and Michael Nagar. “Gastrointestinal manifestations of amyloidosis.” In: *Official journal of the American College of Gastroenterology* 103.3 (2008), pp. 776–787.

[EY36] Carl Eckart and Gale Young. “The approximation of one matrix by another of lower rank.” In: *Psychometrika* 1.3 (1936), pp. 211–218.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks.” In: *International Conference on Machine Learning*. 2017, pp. 1126–1135.

[Fat+15] Mirek Fatyga et al. “A voxel-by-voxel comparison of deformable vector fields obtained by three deformable image registration algorithms applied to 4DCT lung studies.” In: *Frontiers in oncology* 5 (2015), p. 17.

[FHH22] Fenja Falta, Lasse Hansen, and Mattias P Heinrich. “Learning Iterative Optimisation for Deformable Image Registration of Lung CT with Recurrent Convolutional Networks.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2022, pp. 301–309.

[FM02] Bernd Fischer and Jan Modersitzki. “Fast diffusion registration.” In: *Contemporary Mathematics* 313 (2002), pp. 117–128.

[FM03] Bernd Fischer and Jan Modersitzki. "Curvature based image registration." In: *Journal of Mathematical Imaging and Vision* 18.1 (2003), pp. 81–85.

[Fra+18] L Franceschi et al. "Bilevel Programming for Hyperparameter Optimization and Meta-Learning." In: *International Conference on Machine Learning*. 2018, pp. 1563–1572.

[Fuc+20] Fabian Fuchs et al. "Se (3)-transformers: 3d roto-translation equivariant attention networks." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1970–1981.

[Gal+16] Manuel Galli et al. "Machine learning approaches in MALDI-MSI: clinical applications." In: *Expert review of proteomics* 13.7 (2016), pp. 685–696.

[Gap+11] Susan M Gapstur et al. "Association of alcohol intake with pancreatic cancer mortality in never smokers." In: *Archives of internal medicine* 171.5 (2011), pp. 444–451.

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *International Conference on Artificial Intelligence and Statistics*. 2010, pp. 249–256.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[Geh+17] Jonas Gehring et al. "Convolutional Sequence to Sequence Learning." In: *International Conference on Machine Learning*. 2017, pp. 1243–1252.

[Ger+15] Morie A Gertz et al. "Diagnosis, prognosis, and therapy of transthyretin amyloidosis." In: *Journal of the American College of Cardiology* 66.21 (2015), pp. 2451–2466.

[GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks." In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 6645–6649.

[Goo+14] Ian Goodfellow et al. "Generative adversarial nets." In: *Advances in neural information processing systems* 27 (2014).

[Goo+20] Ian Goodfellow et al. "Generative adversarial networks." In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[Gra13] Alex Graves. *Generating sequences with recurrent neural networks*. Tech. rep. arXiv:1308.0850, 2013.

[GRM06] Feng Gan, Guihua Ruan, and Jinyuan Mo. "Baseline correction by improved iterative polynomial fitting with automatic threshold." In: *Chemometrics and Intelligent Laboratory Systems* 82.1-2 (2006), pp. 59–65.

[Gro13] Jürgen H. Gross. *Massenspektrometrie*. Springer Spectrum, 2013.

[Had02] Jacques Hadamard. "Sur les problèmes aux dérivées partielles et leur signification physique." In: *Princeton university bulletin* (1902), pp. 49–52.

[Haz13] Bouke PC Hazenberg. “Amyloidosis: a clinical overview.” In: *Rheumatic Disease Clinics* 39.2 (2013), pp. 323–345.

[HDI+11] Dominic Holland, Anders M Dale, Alzheimer’s Disease Neuroimaging Initiative, et al. “Nonlinear registration of longitudinal images and measurement of change in regions of interest.” In: *Medical image analysis* 15.4 (2011), pp. 489–497.

[He+16] Kaiming He et al. “Deep Residual Learning for Image Recognition.” In: *IEEE Conference on Computer Vision and Pattern Recognition.* 2016.

[He+17] Kaiming He et al. “Mask R-CNN.” In: *IEEE International Conference on Computer Vision.* 2017, pp. 2961–2969.

[Hei+12] Mattias P Heinrich et al. “MIND: Modality independent neighbourhood descriptor for multi-modal deformable registration.” In: *Medical image analysis* 16.7 (2012), pp. 1423–1435.

[Hei19] Mattias P Heinrich. “Closing the gap between deep and conventional image registration using probabilistic dense displacement networks.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention.* 2019, pp. 50–58.

[Her+19] Alessa Hering et al. “Enhancing label-driven deep deformable image registration with local distance metrics for state-of-the-art cardiac motion tracking.” In: *Bildverarbeitung für die Medizin 2019.* Springer, 2019, pp. 309–314.

[Her+22] Alessa Hering et al. “Learn2Reg: comprehensive multi-task medical image registration challenge, dataset and evaluation in the era of deep learning.” In: *IEEE Transactions on Medical Imaging* (2022).

[HG09] Haibo He and Edwardo A Garcia. “Learning from imbalanced data.” In: *IEEE Transactions on knowledge and data engineering* 21.9 (2009), pp. 1263–1284.

[HGH19] Alessa Hering, Bram van Ginneken, and Stefan Heldmann. “mlVIRNET: Multilevel variational image registration network.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention.* 2019, pp. 257–265.

[Hil+01] Derek LG Hill et al. “Medical image registration.” In: *Physics in medicine & biology* 46.3 (2001), R1.

[Hin12] Tijmen Tieleman; Geoffry Hinton. *Lecture 6.5 - RMSProp: Neural Networks for Machine Learning.* Tech. rep. University of Toronto, 2012.

[HKY20] Grant Haskins, Uwe Kruger, and Pingkun Yan. “Deep learning in medical image registration: a survey.” In: *Machine Vision and Applications* 31.1 (2020), pp. 1–18.

[HM05] Eldad Haber and Jan Modersitzki. “Beyond mutual information: A simple and robust alternative.” In: *Bildverarbeitung für die Medizin 2005.* Springer, 2005, pp. 350–354.

[HM06] Eldad Haber and Jan Modersitzki. “Intensity gradient based registration and fusion of multi-modal images.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2006, pp. 726–733.

[Hoo+21] Andrew Hoopes et al. “HyperMorph: Amortized Hyperparameter Learning for Image Registration.” In: *International Conference Information Processing in Medical Imaging*. Springer. 2021, pp. 3–17.

[Hos+21] Timothy Hospedales et al. “Meta-learning in Neural Networks: A Survey.” In: *IEEE transactions on pattern analysis and machine intelligence* 44.9 (2021), pp. 5149–5169.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory.” In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[HSS18] Jie Hu, Li Shen, and Gang Sun. “Squeeze-and-excitation networks.” In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7132–7141.

[Hu+18] Yipeng Hu et al. “Weakly-supervised convolutional neural networks for multimodal image registration.” In: *Medical image analysis* 49 (2018), pp. 1–13.

[HXY15] Zhiheng Huang, Wei Xu, and Kai Yu. *Bidirectional LSTM-CRF Models for Sequence Tagging*. Tech. rep. arXiv:1508.01991, 2015.

[HYC01] Sepp Hochreiter, A Steven Younger, and Peter R Conwell. “Learning to learn using gradient descent.” In: *International Conference on Artificial Neural Networks*. 2001, pp. 87–94.

[Ing+17] Paolo Inglese et al. “Deep learning and 3D-DESI imaging reveal the hidden metabolic heterogeneity of cancer.” In: *Chemical science* 8.5 (2017), pp. 3500–3511.

[IS15] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” In: *International Conference on Machine Learning*. 2015, pp. 448–456.

[Jäh05] Bernd Jähne. *Digitale Bildverarbeitung*. Springer-Verlag, 2005.

[Jia+21] Xi Jia et al. “Learning a model-driven variational network for deformable image registration.” In: *IEEE Transactions on Medical Imaging* 41.1 (2021), pp. 199–212.

[Jon+11] Emrys A Jones et al. “Multiple statistical analysis techniques corroborate intratumor heterogeneity in imaging mass spectrometry datasets of myxofibrosarcoma.” In: *PloS one* 6.9 (2011), e24913.

[JSZ+15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. “Spatial transformer networks.” In: *Advances in Neural Information Processing Systems* 28 (2015).

[Kan+14] Christopher Y Kang et al. “Clinical significance of serum COL6A3 in pancreatic ductal adenocarcinoma.” In: *Journal of Gastrointestinal Surgery* 18.1 (2014), pp. 7–15.

[Kan+23] Frederic Kanter et al. “Classification of Pancreatic Ductal Adenocarcinoma Using MALDI Mass Spectrometry Imaging Combined with Neural Networks.” In: *Cancers* 15.3 (2023).

[Kan18] F. Kanter. “Segmentation of Objects in Thermal Camera Images to Improve Temperature Control in Incubators.” Universität zu Lübeck, 2018.

[KB14] Diederik P Kingma and Jimmy Ba. *Adam: A method for stochastic optimization.* Tech. rep. arXiv:1412.6980, 2014.

[KBD17] András P Keszei, Benjamin Berkels, and Thomas M Deserno. “Survey of non-rigid registration tools in medicine.” In: *Journal of digital imaging* 30.1 (2017), pp. 102–116.

[Kha+21] Salman Khan et al. “Transformers in vision: A survey.” In: *ACM Computing Surveys (CSUR)* (2021).

[Kim+17] Yoon Kim et al. “Structured Attention Networks.” In: *International Conference on Learning Representations.* 2017.

[KL22] Frederic Kanter and Jan Lellmann. “A Flexible Meta Learning Model for Image Registration.” In: *International Conference on Medical Imaging with Deep Learning.* 2022, pp. 638–652.

[Kle+10] Stefan Klein* et al. “elastix: a toolbox for intensity-based medical image registration.” In: *IEEE Transactions on Medical Imaging* 29.1 (Jan. 2010), pp. 196–205.

[Kle+14] Oliver Klein et al. “MALDI imaging mass spectrometry: discrimination of pathophysiological regions in traumatized skeletal muscle by characteristic peptide signatures.” In: *Proteomics* 14.20 (2014), pp. 2249–2260.

[Kle+16] Jorg Kleeff et al. “Pancreatic cancer.” In: *Nature reviews Disease primers* 2.1 (2016), pp. 1–22.

[Kle+19] Oliver Klein et al. “MALDI-imaging for classification of epithelial ovarian cancer histotypes from a tissue microarray using machine learning methods.” In: *Proteomics–Clinical Applications* 13.1 (2019), p. 1700181.

[Kle+20] Oliver Klein et al. “Classification of Inflammatory Bowel Disease from Formalin-Fixed, Paraffin-Embedded Tissue Biopsies via Imaging Mass Spectrometry.” In: *Proteomics–Clinical Applications* 14.6 (2020), p. 1900131.

[Kno+20] Florian Knoll et al. “fastMRI: A Publicly Available Raw k-Space and DICOM Dataset of Knee Images for Accelerated MR Image Reconstruction Using Machine Learning.” In: *Radiology: Artificial Intelligence* 2.1 (2020), e190007.

[Köb+14] Martin Köbel et al. “Ovarian carcinoma histotype determination is highly reproducible, and is improved through the use of immunohistochemistry.” In: *Histopathology* 64.7 (2014), pp. 1004–1013.

[Kön+18] Lars König et al. “A matrix-free approach to parallel and memory-efficient deformable image registration.” In: *SIAM Journal on Scientific Computing* 40.3 (2018), B858–B888.

[Kön18] Lars König. “Matrix-free approaches for deformable image registration with large-scale and real-time applications in medical imaging.” PhD thesis. Institute of Mathematics and Image Computing, University of Lübeck, 2018.

[Kri+16] Mark Kriegsmann et al. “Reliable entity subtyping in non-small cell lung cancer by matrix-assisted laser desorption/ionization imaging mass spectrometry on formalin-fixed paraffin-embedded tissue specimens.” In: *Molecular & Cellular Proteomics* 15.10 (2016), pp. 3081–3089.

[Kri09] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images.” University of Tront, 2009.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks.” In: *Advances in neural information processing systems* 25 (2012).

[KW13] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. Tech. rep. arXiv:1312.6114, 2013.

[Lan+09] Thomas Lange et al. “3D ultrasound-CT registration of the liver using combined landmark-intensity information.” In: *International journal of computer assisted radiology and surgery* 4.1 (2009), pp. 79–88.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning.” In: *Nature* 521.7553 (2015), pp. 436–444.

[LeC+12] Yann A LeCun et al. “Efficient backprop.” In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[LeC+21] Jean Le’Clerc Arrastia et al. “Deeply supervised UNet for semantic segmentation to assist dermatopathological assessment of basal cell carcinoma.” In: *Journal of imaging* 7.4 (2021), p. 71.

[LeC+98] Yann LeCun et al. “Gradient-based learning applied to document recognition.” In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[Lee+09] Honglak Lee et al. “Unsupervised feature learning for audio classification using convolutional deep belief networks.” In: *International Conference Advances in Neural Information Processing Systems*. 2009, pp. 1096–1104.

[Lei+09] Barbara D Leinweber et al. “Improved MALDI-TOF imaging yields increased protein signals at high molecular mass.” In: *Journal of the American Society for Mass Spectrometry* 20.1 (2009), pp. 89–95.

[Leu+19] Johannes Leuschner et al. “Supervised non-negative matrix factorization methods for MALDI imaging applications.” In: *Bioinformatics* 35.11 (2019), pp. 1940–1947.

[LGS99] Thomas Martin Lehmann, Claudia Gonner, and Klaus Spitzer. “Survey: Interpolation methods in medical image processing.” In: *IEEE transactions on medical imaging* 18.11 (1999), pp. 1049–1075.

[LH17] I Loshchilov and F Hutter. “SGDR: Stochastic Gradient Descent with Warm Restarts.” In: *International Conference on Learning Representations*. 2017, pp. 1–16.

[LH19] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization.” In: *International Conference on Learning Representations*. 2019.

[Lia+17] Rui Liao et al. “An artificial agent for robust image registration.” In: *AAAI Conference on Artificial Intelligence*. 2017.

[Lit+17] Geert Litjens et al. “A survey on deep learning in medical image analysis.” In: *Medical image analysis* 42 (2017), pp. 60–88.

[Liu+19] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. Tech. rep. arXiv:1907.11692, 2019.

[Liu+20] Liyuan Liu et al. “On the Variance of the Adaptive Learning Rate and Beyond.” In: *International Conference on Learning Representations*. 2020.

[LMP01] John Lafferty, Andrew McCallum, and Fernando CN Pereira. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. 2001.

[Lon+16] Rémi Longuespée et al. “MALDI mass spectrometry imaging: A cutting-edge tool for fundamental and clinical histopathology.” In: *Proteomics–Clinical Applications* 10.7 (2016), pp. 701–719.

[LQH16] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. “Recurrent Neural Network for Text Classification with Multi-task Learning.” In: *International Joint Conference on Artificial Intelligence*. 2016, pp. 2873–2879.

[LS00] Daniel Lee and H Sebastian Seung. “Algorithms for non-negative matrix factorization.” In: *Advances in neural information processing systems* 13 (2000).

[LS99] Daniel D Lee and H Sebastian Seung. “Learning the parts of objects by non-negative matrix factorization.” In: *Nature* 401.6755 (1999), pp. 788–791.

[Mae+97] Frederik Maes et al. “Multimodality image registration by maximization of mutual information.” In: *IEEE transactions on Medical Imaging* 16.2 (1997), pp. 187–198.

[Mar+15] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[MAT18] MATLAB. *version 9.4.0 (R2018a)*. Natick, Massachusetts: The MathWorks Inc., 2018.

[MB03] Giampaolo Merlini and Vittorio Bellotti. “Molecular mechanisms of amyloidosis.” In: *New England Journal of Medicine* 349.6 (2003), pp. 583–596.

[MC20] Tony CW Mok and Albert Chung. “Large deformation diffeomorphic image registration with laplacian pyramid networks.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2020, pp. 211–221.

[MC21a] Tony CW Mok and Albert Chung. “Conditional Deep Laplacian Pyramid Image Registration Network in Learn2Reg Challenge.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2021, pp. 161–167.

[MC21b] Tony CW Mok and Albert Chung. “Conditional Deformable Image Registration with Convolutional Neural Network.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2021, pp. 35–45.

[McC+05] Gregor McCombie et al. “Spatial and spectral correlations in MALDI mass spectrometry images by clustering and multivariate analysis.” In: *Analytical chemistry* 77.19 (2005), pp. 6118–6124.

[Med+12] Stephan Meding et al. “Tumor classification of six common cancer types based on proteomic profiling by MALDI imaging.” In: *Journal of proteome research* 11.3 (2012), pp. 1996–2003.

[Mei+16] Ivo Meinhold-Heerlein et al. “The new WHO classification of ovarian, fallopian tube, and primary peritoneal cancer and its clinical implications.” In: *Archives of gynecology and obstetrics* 293.4 (2016), pp. 695–700.

[MFP00] Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. “Maximum entropy Markov models for information extraction and segmentation.” In: *International Conference on Machine Learning*. 2000, pp. 591–598.

[MHG+14] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. “Recurrent models of visual attention.” In: *Advances in neural information processing systems* 27 (2014).

[Mit+21] Paul Mittal et al. “Cancer Tissue Classification Using Supervised Machine Learning Applied to MALDI Mass Spectrometry Imaging.” In: *Cancers* 13.21 (2021), p. 5388.

[Mod03] Jan Modersitzki. *Numerical methods for image registration*. OUP Oxford, 2003.

[Mod09] J. Modersitzki. *FAIR: Flexible Algorithms for Image Registration*. SIAM, 2009.

[MSK16] Seiji Mabuchi, Toru Sugiyama, and Tadashi Kimura. “Clear cell carcinoma of the ovary: molecular insights and future therapeutic perspectives.” In: *Journal of gynecologic oncology* 27.3 (2016).

[NKV19] Marc Niethammer, Roland Kwitt, and Francois-Xavier Vialard. “Metric learning for image registration.” In: *IEEE Conference on Computer Vision and Pattern Recognition.* 2019, pp. 8463–8472.

[Nor+07] Jeremy L Norris et al. “Processing MALDI mass spectra to improve mass spectral direct tissue analysis.” In: *International journal of mass spectrometry* 260.2-3 (2007), pp. 212–221.

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* second. New York, NY, USA: Springer, 2006.

[NZY21] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. “A review on the attention mechanism of deep learning.” In: *Neurocomputing* 452 (2021), pp. 48–62.

[Ols+10] Sara H Olson et al. “Allergies, obesity, other risk factors and survival from pancreatic cancer.” In: *International journal of cancer* 127.10 (2010), pp. 2412–2419.

[Pap08] Nils Papenberg. “Ein genereller Registrierungsansatz mit Anwendung in der navigierten Leberchirurgie.” PhD thesis. Institute of Mathematics, University of Lübeck, 2008.

[Pas+19] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” In: *Advances in Neural Information Processing Systems 32.* Curran Associates, Inc., 2019, pp. 8024–8035.

[PDE18] Jaime Prat, Emanuela D’Angelo, and Iñigo Espinosa. “Ovarian carcinomas: at least five different diseases with distinct histological features and molecular genetics.” In: *Human pathology* 80 (2018), pp. 11–27.

[Ped+11] Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python.” In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[Pel+14] Claudio Pelucchi et al. “Smoking and body mass index and survival in pancreatic cancer patients.” In: *Pancreas* 43.1 (2014), pp. 47–52.

[Pli+08] William Plishker et al. “Towards systematic exploration of tradeoffs for medical image registration on heterogeneous platforms.” In: *IEEE Conference Biomedical Circuits and Systems.* 2008, pp. 53–56.

[PO19] Eunbyung Park and Junier B Oliva. “Meta-Curvature.” In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 3314–3324.

[Pol+16] Thomas Polzin et al. “Memory efficient LDDMM for lung CT.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention.* 2016, pp. 28–36.

[Qiu+22] Huaqi Qiu et al. “Embedding Gradient-Based Optimization in Image Registration Networks.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention.* 2022, pp. 56–65.

[Qua+16] Anne S Quante et al. “Projections of cancer incidence and cancer-related deaths in Germany by 2020 and 2030.” In: *Cancer medicine* 5.9 (2016), pp. 2649–2656.

[Rah+14] Lola Rahib et al. “Projecting cancer incidence and deaths to 2030: the unexpected burden of thyroid, liver, and pancreas cancers in the United States.” In: *Cancer research* 74.11 (2014), pp. 2913–2921.

[Ram+21] Aditya Ramesh et al. “Zero-Shot Text-to-Image Generation.” In: *International Conference on Machine Learning*. 2021, pp. 8821–8831.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2015, pp. 234–241.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors.” In: *Nature* 323.6088 (1986), pp. 533–536.

[RL11] Michael Rosenzweig and Heather Landau. “Light chain (AL) amyloidosis: update on diagnosis and management.” In: *Journal of Hematology & Oncology* 4.1 (2011), pp. 1–8.

[RM03] Torsten Rohlfing and Calvin R Maurer. “Nonrigid image registration in shared-memory multiprocessor environments with application to brains, breasts, and bees.” In: *IEEE transactions on information technology in biomedicine* 7.1 (2003), pp. 16–25.

[Roh+17] Marc-Michel Rohé et al. “SVF-Net: learning deformable image registration using shape matching.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2017, pp. 266–274.

[Rue+99] Daniel Rueckert et al. “Nonrigid registration using free-form deformations: application to breast MR images.” In: *IEEE transactions on medical imaging* 18.8 (1999), pp. 712–721.

[San+19] Robin Sandkühler et al. “Recurrent registration neural networks for deformable image registration.” In: *Advances in Neural Information Processing Systems* 32 (2019).

[SC08] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[Sch+18] Jo Schlemper et al. “Attention-Gated Networks for Improving Ultrasound Scan Plane Detection.” In: *Medical Imaging with Deep Learning*. 2018.

[SDP13] Aristeidis Sotiras, Christos Davatzikos, and Nikos Paragios. “Deformable medical image registration: A survey.” In: *IEEE transactions on medical imaging* 32.7 (2013), pp. 1153–1190.

[Sha+10] Ramtin Shams et al. “A survey of medical image registration on multicore and the GPU.” In: *IEEE signal processing magazine* 27.2 (2010), pp. 50–60.

[She+19] Zhengyang Shen et al. “Networks for joint affine and non-parametric image registration.” In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4224–4233.

[Shi+13] Soo J Shin et al. “Unexpected gain of function for the scaffolding protein plectin due to mislocalization in pancreatic cancer.” In: *Proceedings of the National Academy of Sciences* 110.48 (2013), pp. 19414–19419.

[Shi+15] Xingjian Shi et al. “Convolutional LSTM network: A machine learning approach for precipitation nowcasting.” In: *Advances in neural information processing systems* 28 (2015).

[Smi17] Leslie N Smith. “Cyclical learning rates for training neural networks.” In: *IEEE Winter Conference on Applications of Computer Vision*. 2017, pp. 464–472.

[SSP+03] Patrice Y Simard, David Steinkraus, John C Platt, et al. “Best practices for convolutional neural networks applied to visual document analysis.” In: *International Conference on Document Analysis and Recognition*. 2003, pp. 958–962.

[Sto+01] M Stoeckli et al. “Imaging mass spectrometry: a new technology for the analysis of protein expression in mammalian tissues.” In: *Nature medicine* 7.4 (2001), pp. 493–496.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks.” In: *International Conference Advances in Neural Information Processing Systems*. 2014, pp. 3104–3112.

[Svo+20] Christos Svoronos et al. “Prognostic value of COL6A3 in pancreatic adenocarcinoma.” In: *Annals of Hepato-biliary-pancreatic Surgery* 24.1 (2020), p. 52.

[SWS17] Dinggang Shen, Guorong Wu, and Heung-Il Suk. “Deep learning in medical image analysis.” In: *Annual review of biomedical engineering* 19 (2017), pp. 221–248.

[SZ15] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” In: *International Conference on Learning Representations*. 2015.

[TBU00] Philippe Thévenaz, Thierry Blu, and Michael Unser. “Image interpolation and resampling.” In: *Handbook of medical imaging, processing and analysis* 1.1 (2000), pp. 393–420.

[Tel+21] Akbar Telikani et al. “Evolutionary machine learning: A survey.” In: *ACM Computing Surveys (CSUR)* 54.8 (2021), pp. 1–35.

[Thi98] J-P Thirion. “Image matching as a diffusion process: an analogy with Maxwell’s demons.” In: *Medical image analysis* 2.3 (1998), pp. 243–260.

[Tho+18] Nathaniel Thomas et al. *Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds*. Tech. rep. Google, 2018.

[Vas+17] Ashish Vaswani et al. "Attention is All you Need." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 5998–6008.

[VCV20] Nico Verbeeck, Richard M Caprioli, and Raf Van de Plas. "Unsupervised machine learning for exploratory data analysis in imaging mass spectrometry." In: *Mass spectrometry reviews* 39.3 (2020), pp. 245–291.

[Vel+17] Petar Veličković et al. "Graph Attention Networks." In: *International Conference on Learning Representations*. 2017.

[Ves+14] Kirill A Veselkov et al. "Chemo-informatic strategy for imaging mass spectrometry-based hyperspectral profiling of lipid signatures in colorectal cancer." In: *Proceedings of the National Academy of Sciences* 111.3 (2014), pp. 1216–1221.

[VG96] Charles F Van Loan and G Golub. "Matrix computations (Johns Hopkins studies in mathematical sciences)." In: *Matrix Computations* 53 (1996).

[Vir+20] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." In: *Nature Methods* 17 (2020), pp. 261–272.

[Vos+17] Bob D de Vos et al. "End-to-end unsupervised deformable image registration with a convolutional neural network." In: *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2017, pp. 204–212.

[WAD11] Jeramie D Watrous, Theodore Alexandrov, and Pieter C Dorrestein. "The evolving field of imaging mass spectrometry and its impact on future biological research." In: *Journal of Mass Spectrometry* 46.2 (2011), pp. 209–222.

[Wal+08] Axel Walch et al. "MALDI imaging mass spectrometry for direct tissue analysis: a new frontier for molecular histology." In: *Histochemistry and cell biology* 130.3 (2008), pp. 421–434.

[Win+17] Martin Winter et al. "MALDI mass spectrometry imaging: a novel tool for the identification and classification of amyloidosis." In: *Proteomics* 17.22 (2017), p. 1700236.

[Wu+21] Zhiyang Wu et al. "Discovery of Spatial Peptide Signatures for Neuroblastoma Risk Assessment by MALDI Mass Spectrometry Imaging." In: *Cancers* 13.13 (2021), p. 3184.

[Xu+15] Kelvin Xu et al. "Show, attend and tell: Neural image caption generation with visual attention." In: *International Conference on Machine Learning*. 2015, pp. 2048–2057.

[Xu+21] Junshen Xu et al. "Multi-scale neural odes for 3d medical image registration." In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2021, pp. 213–223.

[Yan+15] Jian Bo Yang et al. “Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition.” In: *International Conference on Artificial Intelligence*. 2015, pp. 3995–4001.

[Yan+17] Xiao Yang et al. “Quicksilver: Fast predictive image registration–a deep learning approach.” In: *NeuroImage* 158 (2017), pp. 378–396.

[Yao+17] Jiawen Yao et al. “Deep Correlational Learning for Survival Prediction from Multi-modality Data.” In: *International Conference Medical Image Computing and Computer-Assisted Intervention*. Springer, 2017, pp. 406–414.

[Zbo+18] Jure Zbontar et al. *fastMRI: An Open Dataset and Benchmarks for Accelerated MRI*. Tech. rep. arXiv:1811.08839, 2018.

[Zha+19a] Zhiwen Zhang et al. “Upregulation of nucleoprotein AHNAK is associated with poor outcome of pancreatic ductal adenocarcinoma prognosis via mediating epithelial-mesenchymal transition.” In: *Journal of Cancer* 10.16 (2019), p. 3860.

[Zha+19b] Shengyu Zhao et al. “Recursive cascaded networks for unsupervised medical image registration.” In: *IEEE International Conference on Computer Vision*. 2019, pp. 10600–10610.

[Zha+20] Wanqiu Zhang et al. “Spatially-Aware Clustering of Ion Images in Mass Spectrometry Imaging Data Using Deep Learning.” In: *bioRxiv* (2020).

[Zhe+14] Yi Zheng et al. “Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks.” In: *International Conference Web-Age Information Management*. 2014, pp. 298–310.

[Zho+16] Bolei Zhou et al. “Learning Deep Features for Discriminative Localization.” In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2921–2929.