INAUGURAL – DISSERTATION

zur

Erlangung der Doktorwürde

der

Gesamtfakultät für Mathematik, Ingenieur- und Naturwissenschaften

der

Ruprecht – Karls – Universität

Heidelberg

vorgelegt von

Bitto, Verena, Dipl.-Ing.

aus Wels, Österreich

Tag der mündlichen Prüfung:

Enhancing Biomarker Discovery in Tumor Hypoxia for Head and Neck Squamous Cell Carcinoma: Advancing Spatial Omics Data Accessibility through Convolutional Autoencoders

Betreuer: Prof. Dr. Klaus H. Maier-Hein

Prof. Dr. Benedikt Brors

Abstract

Spatial omics data shows potential to reveal novel insights into the underlying mechanisms of cancer. Yet the high-dimensional and highly correlated feature space imposes challenges on analysis. In this thesis, the implementation of convolutional autoencoders to extract explainable features for biomarker discovery is examined, exemplified on tumor hypoxia.

Mass spectrometry imaging and spatial transcriptomics experiments were performed on consecutive tissue slices of head and neck squamous cell carcinoma tumor models. To advance accessibility of these spatial omics modalities, data was reduced by convolutional autoencoders and the resulting latent space features were ranked for association with tumor hypoxia through random forest feature importance measures. With the help of a newly proposed recovery method, the contribution of original features to a latent feature was derived, thereby retaining biological relevant information. The derived genes and peptides were compared against the ranked genes and peptides of a random forest only model. The feature sets of the autoencoder approaches achieved consistently higher scores when evaluated using the structural similarity index measure. In contrast, the features of the random forest only models contained many more noisy hypoxia associations caused by the multicollinearity of features.

Several promising unimodal and multimodal biomarker candidates of mass spectrometry imaging and spatial transcriptomics data for tumor hypoxia were identified. Multimodal biomarkers were identified through correlation analysis of aligned serial tissue slices from both spatial omics modalities in four samples. For a more elaborate integration, it was outlined how the molecular information of multiple spatial omics modalities may be combined without error-prone alignment of consecutive tissue slices. Instead, the spatial omics modalities may be learned directly from the readily available microscopy images using convolutional neural networks. Then, the learned molecular information may be predicted from microscopy images of other spatial omics modalities. Preliminary results demonstrated that the learning of the latent space features of autoencoders yielded more

accurate predictions than when learning was performed on the raw and sparse spatial omics features. However, it necessitates further investigation whether also hypoxia-associated features can be acquired accurately from microscopy images.

Overall, the findings show that convolutional autoencoders accompanied by random forest models retain more biological relevant information for biomarker discovery than without prior feature extraction. Considering the increasing amount of available (spatial) omics data, deep learning feature extraction will become evermore important. This thesis contributes to the overall understanding of autoencoders by showcasing how specific characteristics in spatial omics data reflect in the latent space and how they can be addressed through hyperparameter configurations.

Zusammenfassung

Räumlich aufgelöste Omics-Daten könnten einen wichtigen Beitrag dazu leisten, bisher unbekannte Mechanismen von Krebs zu erforschen. Allerdings ist die Analyse von diesen hochdimensionalen und zugleich stark korrelierenden Daten schwierig. Diese Arbeit untersucht, inwiefern Convolutional Autoencoder erklärbare Features extrahieren können, die zur Identifikation neuer Biomarker für Tumor Hypoxie genutzt werden können. Erklärbar bedeutet in diesem Zusammenhang, dass eine Verknüpfung mit den ursprünglichen Features ermöglicht werden soll, um den molekularen Kontext zu erhalten.

Für diese Arbeit wurden Mass Spectrometry Imaging und Spatial Transcriptomics Experimente auf konsekutiven Tumorschnitten mehrerer Kopf-Hals-Karzinom-Modelle durchgeführt. Diese räumlich aufgelösten Omics-Daten wurden mittels Convolutional Autoencoder kodiert und die resultierenden latenten Features wurden auf Assoziationen mit Hypoxie überprüft. Die Relevanz der Assoziationen wurden mittels Feature Importance Metriken von Random Forest Modellen bestimmt. Eine von mir neu entwickelte Methode erlaubt dabei, den Beitrag aller ursprünglichen Features auf ein latentes Feature abzuschätzen. Die dadurch identifizierten Gene und Peptide wurden mit jenen verglichen, die aus reinen Random Forest Modellen abgeleitet werden können. Die Features, die durch den Autoencoder gewonnen wurden, wiesen dabei eine konsistent höhere Ähnlichkeit zueinander auf (gemäß dem Index für strukturelle Ähnlichkeit), als die Features aus den alleinigen Random Forest Modellen. Die Features der Random Forest only Modelle führten dabei zugleich zu deutlich mehr falsch-positiven Assoziationen zu Hypoxie, was vermutlich auf die Multikollinearität der Features zurückzuführen ist.

Mehrere unimodale und multimodale Biomarker-Kandidaten für Hypoxie wurden aus Mass Spectrometry Imaging und Spatial Transcriptomics Daten abgeleitet. Die multimodalen Biomarker-Kandidaten wurden mittels Korrelationsanalyse von alignierten konsekutiven Tumorschnitten identifiziert. Im letzten Teil dieser Arbeit wurde untersucht, ob Deep Learning Modelle für die Integration der Omics-Daten, als Alternative zu der

fehleranfälligen Co-Registrierung von konsekutiven Schnitten, genutzt werden können. Ziel hierbei ist, molekulare Informationen auf Basis von Mikroskopbilder zu erlernen und auf Mikroskopbildern anderer (Omics-)Experimente anzuwenden. Im Rahmen dieser Arbeit wurde gezeigt, dass das Erlernen von Peptidinformationen aus Mikroskopbildern grundsätzlich möglich ist, insbesondere dann, wenn anstelle der Rohdaten, die extrahierten Features des Autoencoders verwendet wurden. Allerdings sind weitere Untersuchungen notwendig um herauszufinden, inwiefern sich auch Peptide, die mit Hypoxie assoziiert sind, durch vorhandene Strukturen in Mikroskopbilder erlernen lassen.

Die Ergebnisse dieser Thesis zeigen, dass Convolutional Autoencoder in Kombination mit Random Forest Modellen zuverlässigere biologische Informationen extrahieren können als ohne vorausgegangene Reduktion der Daten. Berücksichtigt man, dass die Anzahl und Menge an Omics-Daten weiter steigen wird, so ist es naheliegend, dass Methoden zur Feature Extraktion weiter an Relevanz gewinnen werden. Diese Arbeit trägt zum allgemeinen Verständnis von Autoencodern bei, indem gezeigt wird, wie sich bestimmte Datencharakteristiken auf die latenten Features auswirken können, beziehungsweise, wie Hyperparameter konfiguriert werden müssen um erklärbare Features zu extrahieren.

Acknowledgements

First, I would like to thank my supervisors, Klaus Maier-Hein, Benedikt Brors, Michael Baumann and Ina Kurth, for giving me the opportunity to engage in the topic of computational oncology, despite coming from a completely different background. I really mean it! I am grateful for their guidance and support throughout my PhD. I want to thank Klaus for helping me to find the right balance between computer science, medicine and biology. To Benedikt, I am grateful for staying calm and confident when I started to doubt. Despite being challenging, I cherish the memory of Ina, Michael and me working together for hours to end my very first paper in this field - I gained so many valuable insights from it! Also, I want to thank Ina for always being there to listen – you have been incredibly supportive! A big thank you also extends to Pia Hönscheid, for answering all my questions about mass spectrometry, even when I asked them repeatedly. I always enjoyed our video calls! My appreciation also goes to Denis Schapiro, for the discussion on the registration of consecutive tissue slices and the exchange on utilizing the capabilities of deep leaning models instead.

With deep gratitude, I would like to thank my grandmother, for consistently having faith in my ability to accomplish anything I aspire to. Without her profound trust and support, I would not be the person I am today. A heartfelt thank you is extended to my dear friend Freddy, the one whom I discussed the initial steps towards a return to academia, and who has accompanied me on this wonderful journey, eagerly listening to every detail of my work. A warm thank you goes to my family and their support, in particular to my father who recognized art in H&E stains.

I am grateful to have been a fellow of Helmholtz Information & Data Science School for Health (HIDSS4Health). Without the school, the transition from industry to academia would have been challenging for me.

Finally, I want to thank my dear colleagues: María José Besso, Mareike Roscher, Cristina

Conde Lopez, Rose Euer-Lange, Wahyu Wijaya Hadiwikarta, Safayat Mahmud Khan, Julian Schlecker, Lena Voith von Voithenberg, Corinna Sprengart, Ferdi Popp, Nick Abad, Julia Münch, Alexandra Walter, Paul Maria Scheikl, Julian Herold, Elaine Zaunseder. Thank you all for the shared moments, the many laughs, for the grounding! A big thanks to María and Christian Sperling, for carrying out the wet lab experiments and their patience when adjustments had to be made. I would also like to thank María (again!), as well as Ina and Nick for checking this work for typos. Lastly, I would like to thank Melanie Schellenberg, who kept me from going crazy in the final phase of my PhD and who provided me with numerous invaluable pieces of advice!

Contents

List of Figures

List of Tables

List of Code Snippets

1 Introduction

Spatial omics data promises to reveal novel insights into the molecular biology of tumor heterogeneity in cancer. Yet for the discovery of biological markers, this data is rarely used because it is difficult to analyze (high dimensionality, sparsity, multicollinearity of features, underlying molecules may remain obscured in some technologies). Therefore, this thesis proposes a framework to make spatial omics data more accessible using convolutional autoencoders.

The focus of this thesis is to identify biological markers for tumor hypoxia in head and neck squamous cell carcinoma (HNSCC) using mass spectrometry imaging (MSI) and spatial transcriptomics (SPT) data. Tumor hypoxia is a dynamic state of reduced oxygen levels within a tumor that is associated with bad prognosis [1]. Many different biological markers, or so-called biomarkers, which indicate whether a tumor of a patient is hypoxic or not, were proposed in HNSCC. However, so far, existing biomarkers showed conflicting findings in predicting treatment outcomes when evaluated on independent patient cohorts [2, 3]. The majority of proposed biomarkers in HNSCC is unimodal, relying either on imaging or solely on molecular data. Spatial omics data, however, allows both to be considered, the molecular information and its spatial context. Thus, I hypothesize that spatial omics data allows us to establish more reliable biomarkers. The data investigated in this thesis includes fragmented peptide information from MSI and gene expression data from SPT derived from HNSCC tumor models. In a first approach, gene expression and peptide information data are analyzed separately. The later chapter of this thesis considers the combination of both spatial omics data.

The analysis of the spatial omics data is primarily carried out using convolutional autoencoders. This type of machine learning algorithm reduces high-dimensional data into a

lower-dimensional representation. Technically, a convolutional autoencoder (CAE) consists most commonly of two interconnected neural networks: an encoder and a decoder [4]. The use of neural networks has been steadily increasing in healthcare applications in the last years, especially in the field of medical imaging. In this domain, neural networks are used for disease classification or tissue segmentation, among others [5]. However, in other medical areas like genomics, it is more relevant to understand the underlying biology of a disease to be able to identify potential biomarkers or targets [6]. Here, machine learning methods are still underrepresented as results are often difficult to understand and interpret for humans. Therefore, the aim of this thesis is to utilize CAEs such that the molecular information of hypoxic tumor regions is still accessible. It is shown that CAEs can complement methodologically less complex random forest (RF) models in a biomedical meaningful manner.

1.1 Contributions

The increasing feature space of (spatial) omics data makes traditional statistical tests infeasible and therefore requires novel strategies for analysis. In this thesis, it is investigated how CAEs can be utilized to reduce the high-dimensional space of spatial omics data while at the same time allowing for explainable results. MSI data and SPT data from consecutive tissue slices of HNSCC xenograft models are utilized. In the Chapter **Convolutional Autoencoders for Aggregating Spatial Omics Data to Derive Hypoxia-Associated Biomarkers**, the following contributions are made:

- CAEs combined with RF models are used to extract and identify features associated with hypoxia.
- More generally, it is shown that CAEs retain low intensity features of spatial omics data if the loss function and hyperparameters are configured accordingly.
- A recovery method is presented that identifies which original features contributed to the latent features of the trained CAEs. This tracks back the molecular information.
- The identified features of the combined convolutional autoencoder and random forest (CAERF) approach are compared against features from random forest only models, highlighting that more biologically meaningful features are extracted by

means of CAERF.

- A semi-supervised CAE approach is presented which incorporates hypoxia labels and thereby reduces noisy associations and increases the validity of results.
- Challenges of specific characteristics in different spatial omics modalities for CAEs are described and addressed.
- Several peptide and gene candidates as potential biomarkers for tumor hypoxia in HNSCC tumor models are presented.

Different spatial omics data presents different challenges, which are covered in subsection **Mass Spectrometry Imaging** and subsection **Spatial Transcriptomics**.

For **Mass Spectrometry Imaging**, addressed challenges include:

- the variations in mass-to-charge ratio (m/z) values from different samples in mass spectrometry (MS). MS instruments typically output m/z values and not the underlying peptides. It is highlighted that AEs can be used to aggregate related m/z values without the need for heavy pre-processing.
- the heuristic mapping of m/z values of MSI to more precise masses from liquid chromotography (LC)-MS/MS to identify peptide candidates for tumor hypoxia. This is necessary as the mass accuracy and mass resolving power of MSI is too low for directly inferring peptides.

For **Spatial Transcriptomics**, addressed challenges include:

- the "orange crate packing" arrangement of spatial spots. It is shown how to adjust the hyperparameters and the loss function of CAEs to train on the SPT data accordingly.

In Chapter **Combining Spatial Omics Data To Identify More Robust Biomarkers**, contributions involve:

- Serial slices of MSI and SPT data from the same xenograft sample are combined to identify multimodal gene-peptide biomarkers.
- The impact of different spatial resolutions in different modalities for data integration

is discussed.

- It is investigated if hypoxia-associated peptide information of MSI data can be reliably learned from hematoxylin and eosin (H&E) images using a convolutional neural network (CNN). As part of future work, I consider using the trained CNN to predict MSI data on top of SPT data.
- It is investigated if dimensionality-reduced, hypoxia-associated peptide information of MSI data can be reliably acquired from H&E images using a CNN. It is shown that the latent space of the CAE had denoising capabilities and allowed for more accurate predictions than without prior feature extraction.

1.2 Structure

The thesis is structured as follows: Chapter 2 covers the background of HNSCC and known biomarkers for tumor hypoxia. It further describes the characteristics of (spatial) omics data and feature selection / extraction methods commonly applied for the analysis of high-dimensional data. Chapter 3 summarizes the used data and software. In Chapter 4, the CAERF approach for the analysis of spatial omics data is presented and compared against RF models. Chapter 5 focuses on the combination of spatial omics data. Chapter 6 concludes this thesis with an outline of limitations, potential improvements and extensions for the proposed CAERF approach and the combination of spatial omics data.

1.3 Terminology

In omics data, *features* correspond to the molecular information like genes or m/z values. The *observations* or *samples* are typically the individuals, e.g., patients or xenografts, used to measure this molecular information, or the materials derived from them. Specifically, the observations or samples in spatial omics are pixels or patches, i.e., small regions of the data, derived from one or multiple individuals.

2 Background

This chapter introduces the relevant biomedical aspects of head and neck squamous cell carcinoma (HNSCC), tumor hypoxia and the field of biomarker discovery in cancer. The (spatial) omics modalities used in this thesis to derive novel markers are described. Then, the need for novel biomarkers for tumor hypoxia is illustrated by discussing the limitations of existing ones in HNSCC. The chapter concludes with a brief summary of relevant methods for data analysis and their challenges.

2.1 Head and Neck Squamous Cell Carcinoma

HNSCC is one of the most common cancer wordwide with around 880,000 new cases and 450,000 deaths in 2020 ([7], considering cancer sites "hypopharynx", "larynx', "lip, oral cavity", "nasopharynx" and "oropharynx"). In Germany, patients with HNSCC are usually treated with postoperative radio(chemo)therapy or, in case of functionally inoperable tumors, with primary radiochemotherapy (RCTx), i.e., a combinational treatment of radiotherapy and chemotherapy [8]. Historically, HNSCC was treated with surgery or radiotherapy, with first trials starting to evaluate the addition of chemotherapy in terms of platinum compounds (cisplatin or carboplatin) in the 1980s [9, 10]. The efficacy of clinical interventions is measured by comparison of so-called endpoints. One endpoint considered as gold standard in many clinical trials is overall survival (OS) of patients, defined as time from randomization in a clinical trial until death from any cause [11]. However, to evaluate more specific effects of treatment, usually additional endpoints such as disease-free survival (DSF) or event-free survival (EFS) are inspected [12]. In solid tumors, loco-regional failure (LRF), i.e., the time from randomization to the first loco-regional relapse, is commonly used as primary or secondary endpoint [13]. The

addition of chemotherapy to radiotherapy was found to significantly decrease LRF and significantly increase EFS in HNSCC [14]. Since then, RCTx is the standard treatment for inoperable HNSCC in Germany. However, treatment success has shown to rely on many factors like sex, age or tumor specific characteristics. In HNSCC, the infection status for the Human Papillomavirus (HPV) has been recognized as a prognostic marker [15, 16]. HPV-positive tumors are considered to be more radiosensitive [17]. Therefore, it is investigated whether a reduction of radiation dosage might lead to less side-effects in patients while preserving loco-regional tumor control (LRC) rates [18]. One of the tumor characteristics associated with bad prognosis is tumor hypoxia, a state of low oxygen levels in solid tumors [1]. Particularly, hypoxia is also associated with resistance to radio- and chemotherapy [19]. Therefore, hypoxia has been considered as a parameter for treatment adjustments for many decades [20]. However, compared to the HPV status, it is more difficult to identify to which degree a tumor is hypoxic. Different approaches to assess tumor characteristics in general, and hypoxia specifically have been proposed, which are discussed in the following.

2.2 Biomarker Discovery

Cancer biomarkers, i.e., biological markers, are indicators of risk for cancer occurrence, for poor response to therapy or for bad outcome [21]. In this thesis, biomarkers are considered in the context of tumor hypoxia, i.e., markers that indicate the risk for treatment failure, which can be attributed to hypoxic regions within a tumor.

2.2.1 Molecular Biomarkers

Molecular cancer biomarkers measure biomolecules which are present or produced in cancer tissue [21]. For detection of such markers, most commonly tumor tissue is utilized either from biopsies of patients or by means of so-called xenograft models, in which small human tumor pieces are transplanted into mice [22]. Generally, one can distinguish between non-targeted and targeted sequencing approaches to measure biomolecules of interest. While the former technologies are capable of measuring tens of thousands of molecules, the latter are limited to a pre-defined set of molecules. For the topic of biomarker discovery, targeted approaches are not discussed here. The following subsections describe the broader context of the genomic and proteomic data used in this thesis. Other molecular biomarkers

like DNA or miRNA biomarkers are not touched.

2.2.1.1 Transcriptomics

The transcriptome is considered as the total number of ribonucleic acid (RNA) transcripts of a cell, measured at a specific point in time [23]. In technologies like microarray hybridization or RNA-sequencing, the transcriptome of a bulk of cells (e.g., from an entire tissue slice) is captured and combined. This is problematic as tumors can display significant levels of heterogeneity, which cannot be captured with bulk approaches [24]. Single cell RNA sequencing (scRNA) overcomes these limitations by capturing transcripts per cell. Still, the spatial arrangement of a tumor, and thus its spatial heterogeneity, is not captured by means of scRNA. Spatial transcriptomics bridges this gap by retaining the spatial information of tissue and was nominated method of the year in 2020 by *Nature Methods* [25]. At the moment, spatial transcriptomics technologies are not yet readily available in single cell resolution for non-targeted approaches [26], but methods are evolving. Many spatial transcriptomics technologies have been developed, which can be categorized by the underlying methods (e.g., fluorescence in situ hybridization or sequencing-based methods) and the number of detectable transcripts [27].

2.2.1.2 Proteomics

Proteins, i.e., macromolecules composed of peptides, are of special interest for biomarker discovery, as they provide more precise insights into the cellular state compared to RNA [28]. One way to measure proteins is by means of mass spectrometry (MS). A mass spectrometer comprises an ion source, a mass analyzer and a detector [29]: (1) The ion source produces charged analytes (e.g., peptide ions). (2) The ions are separated by their mass-to-charge ratio (m/z) by the mass analyzer. (3) The abundance of ions at different m/z values, referred to as mass spectrum, is measured by the detector. The ionization of proteins, however, is challenging, as relatively large molecules are easily destroyed during the process. These problems were overcome with electrospray ionization (ESI) and matrix-assisted laser desorption ionization (MALDI) [30]. In terms of the latter, ionization is achieved by means of a specific matrix which is applied onto the tissue, depending upon the molecule of interest, such as the α-cyano-4-hydroxycinnamic acid (CHCA) matrix for ionizing peptides [31]. Additionally, enzymatic digestion like trypsin is often part of sample preparation to break proteins into smaller peptides, enhancing

ionization efficiency. The protein identification capability of MS, i.e., to link m/z values to actual peptides, is impacted by several factors. For example, the mass analyzer will impact mass accuracy and mass resolving power, with time of flight (TOF)-based approaches lacking sufficient resolution for accurate peptide identification [32]. Newer technologies like Fourier Transform Ion Cyclotron Resonance (FTICR) [33] allow for higher mass resolving powers and mass accuracies [33, 34], but widespread application is currently hindered by the higher costs. Other factors include enzymatic digestion, which facilitates identification as the used enzymes have known cleavage specificity [30]. In MALDI, also the utilized matrix will impact mass accuracy due to ion suppression effects [35]. Given these numerous factors affecting peptide identification, TOF-based MS experiments are often complemented by more precise mass information from tandem mass spectrometry experiments [36], like liquid chromotography (LC)-MS/MS. In tandem mass spectrometry, two mass analyzers are involved: The first mass analyzer separates and isolates an ion of interest, while the second mass analyzer analyzes the resulting fragments [29]. In LC-MS/MS, liquid chromatography is used as pre-processing step for the separation of ions, which reduces the overall complexity of a sample by splitting it into smaller fractions before inputting it into the mass spectrometer [37]. Traditional MS is typically performed on one (or multiple) tissue slice(s) as bulk. Similar to spatial transcriptomics, mass spectrometry imaging (MSI) allows to retain the spatial information of the measured molecules. An often-used visual representation of MSI data is ion images, which illustrate the intensities at a specific m/z value in a spatial context [38]. In the last decade, the spatial resolution of MSI has been consistently improved from $100\mu m$ down to $1.5\mu m$. [39, 40]. One limiting factor for full tissue processing at single cell resolution is the data size, which is expected to reach terabytes to petabytes in the next decade [40]. Other factors which influence the spatial resolution are the utilized laser, the ionization technique, and the type of molecules investigated.

2.2.2 Image-Based Biomarkers

Different imaging methods have been employed to extract features, which can serve as biomarkers. On a macroscopic level, positron emission tomography (PET) imaging is commonly used for the detection and monitoring of cancer progression [41]. PET allows for molecular imaging, i.e., the measurement of biological processes [41] like glucose uptake by means of radiotracers. On a microscopic level, hematoxylin and eosin (H&E) stains are

frequently utilized to confirm the presence of cancer by pathologists [42]. Fluorescence images make use of fluorescent dyes to check for certain antibodies in a tissues and are often complementing H&E stains. Various other imaging technique are employed for biomarker discovery which are not part of this work and hence not described.

2.3 Existing Biomarkers for Tumor Hypoxia in HNSCC

Many different biomarkers to estimate tumor hypoxia have been proposed. The majority of them are unimodal, i.e., they rely on a single source to estimate hypoxia. In the following, special focus is put on biomarkers with prognostic value for RCTx regimes, as it is the standard treatment for inoperable HNSCC in Germany. For clinical studies, primary endpoints LRC (or loco-regional failure [LRF]) and progression-free survival (PFS) are favored over studies considering only OS rates, as these endpoints are more directly connected to therapy success (see Section 2.1).

2.3.1 Molecular Biomarkers

One approach to assess hypoxia is by means of transcript levels of hypoxia-related genes, often simply referred to as gene signatures. For HNSCC, several hypoxia gene signatures have been proposed [43, 44], which have later been refined [45–47]. Of note, there is no standardized way to derive and apply gene signatures [48]. This introduces different sources of variability, which hampers comparability among studies. Among the proposed gene signatures, the hypoxia 15 gene signature by Toustrup et al. has shown to be prognostic for LRC [46]. In that study, tumors of patients were split into being either more or less hypoxic, depending on the pre-treatment transcript levels of 15 genes. However, in later studies, in which patients underwent RCTx rather than radiotherapy alone, the prognostic value could not be confirmed [8, 49, 50]. Overall, this suggests that existing hypoxia gene signatures cover some but not all biological relevant aspects of tumor hypoxia.

On the protein level, HIF1α was investigated as potential marker for hypoxia. This transcription factor, i.e., a protein with gene regulation capabilities, is known to stabilize under hypoxic conditions, leading to changes in regulations of VEGF, CA-IX, GLUT-1 and others [51]. However, later findings suggested, that HIF1α itself is not a suitable

biomarker for hypoxia, specifically under chronic hypoxia [52]. Chronic hypoxia is caused primarily by limitations in oxygen diffusion, while acute hypoxia is defined as temporary disturbance in perfusion [53]. Although some of the mentioned proteins were associated with prognosis in some HNSCC studies, a direct connection to hypoxia is difficult to draw, primarily because many other factors regulate the expression of endogenous markers apart from hypoxia [19, 51]. Adding to the overall complexity, tumor hypoxia, even in the case of chronic hypoxia, can be considered as dynamic state which may change over time [54]. As a consequence, Janssen et al. suggested that multiple markers may be necessary to characterize hypoxia [19].

2.3.2 Image-based Biomarkers

Non-invasive PET imaging approaches can assess hypoxia by various tracers, like ^{18}F-fluoromisonidazole (FMISO) or ^{18}F-fluoroazomycin-arabinoside (FAZA). Several studies found different PET parameters to be prognostic for LRC for patients with HNSCC that received RCTx, conducted either before [55] or during treatment [56]. The variability of FMISO and FAZA tracers is comparably low, such that a common cut-off value to separate patients into risk groups was proposed [55]. However, so far there is no common agreement on which parameters to asses, e.g., the precise reference volume or the time of assessment, which hampers the comparability of results between centers. Additionally, small hypoxic regions may be missed due to the relatively low resolution of one voxel [57]. To date, PET imaging is rarely used for hypoxia assessment in clinical practice, mainly due to its high costs.

Hypoxia can also be estimated from pimonidazole binding levels of tumor biopsy material and was found to be prognostic for LRC in patients with HNSCC [58]. In that work, the fraction of hypoxia was quantified by means of fluorescence images, more precisely by pimonidazole-stained cells. Compared to PET scans, pimonidazole enables a microscopic approximation of hypoxia [57]. However, the necessity for intravenous injection before biopsy makes pimonidazole labeling impractical and error-prone for clinical use.

2.3.3 Biomarkers from Our Pre-Clinical Study

As part of my PhD, I investigated gene expression and histological data from seven HNSCC xenograft models to derive potential biomarkers for hypoxia. The same models were

considered for this thesis and are described in Section 3. In Koi and Bitto et al., we found that the seven tumor models exhibited similar degrees of pre-treatment hypoxia, measured as pimonidazole hypoxic volume [59]. However, only four tumor models (named FaDu, SAS, UT5, UT8) showed a significant decrease in hypoxia during treatment with RCTx alone as well as with a combinational treatment of RCTx and nimorazole. Nimorazole is a hypoxic cell radiosensitizer that is used as part of the standard treatment for inoperable HNSCC in Denmark, but is not approved in Germany (for details see Baumann et al. [60]). Additionally, the tumor control rate (TCR) was evaluated until day 120–190 on the same tumor models for comparison of RCTx plus nimorazole and RCTx alone (control group). For two out of the four tumor models (FaDu, SAS) that showed a decrease in hypoxia, also the TCR after treatment with RCTx plus nimorazole was significantly increased. For the other two models (UT5, UT8), a positive (though not statistically significant) association was found. The remaining three models (named CAL33, UT45, SAT) exhibited no decrease in hypoxia during treatment, and were also lacking any positive effect on TCR after treatment with RCTx plus nimorazole. Evaluating potentially differences between responding (FaDu, SAS) and non-responding (CAL33, UT45, SAT) tumor models, we identified 12 genes which were differentially expressed. When we investigated these genes on data from patients with HNSCC, we found that they were prognostic for LRC in RCTx. This indicated that the regulation of these genes may also play a relevant role in humans. We hypothesized that a higher regulation of the genes might indicate a higher degree of RCTx resistance per se, potentially associated with hypoxia.

2.4 The Need for Dimensionality Reduction

Biomolecular data imposes several challenges on data analyses. First, biomolecular data aims to acquire as much biological signals as possible, leading to thousands of dimensions. Second, data acquisition is time-consuming and often expensive, typically leading to the collection of only a small sample size. Both characteristics (high feature dimensionality, low sample size) make standard statistical approaches infeasible or prone to statistical errors. For example, applying t-tests to omics data often leads to thousands of statistically significant features, even after controlling for the false discovery rate (FDR) [61]. Clearly, using this amount of features for building a model will lead to severe overfitting, given the limited sample size [62]. Moreover, the need for evaluating too many features is

impractical in a clinical setup.

One common strategy to overcome the so-called curse of dimensionality is to reduce dimensions for the sake of losing some biological signals during data pre-processing. The following subsections highlight the relevant concepts and strategies for dimensionality reduction in context of this thesis. In general, methods are either unsupervised, i.e., they work solely on the underlying omics data, or supervised, i.e., they incorporate apart from the omics data itself some kind of labels to choose relevant features. Semi-supervised approaches combine both strategies by using unlabeled as well as labeled data.

2.4.1 Feature Selection

Feature selection methods reduce the original high-dimensional feature space to the most informative features [63]. Often, feature selection is applied as part of a supervised classification or regression task. However, it is important to note that for biomarker discovery a model is not primarily built for prediction. Instead, the objective is to pinpoint features which explain biomolecular mechanisms of a disease. In this context, it is often not sufficient to detect single markers since they might be regulated by diverse biological pathways, as pointed out in the last sections. Rather, one aims to find what Nilsson et al. termed *all relevant* features, which is mathematically a harder problem than identifying a *minimal-optimal* set of features [64]. As a consequence, many feature selection methods are not designed for identifying *all relevant* features. Subsequently, some supervised feature selection methods are described.

2.4.1.1 Filtering

Commonly applied filtering methods include univariate filtering such as t-tests or ANOVA, though as pointed out, they were not specifically designed for omics data. Apart from potential overfitting, these methods do not take into account feature dependencies [65]. In consequence, they will miss features that might be relevant only in conjunction with other features [66]. More advanced multivariate filtering approaches like limma [67] or DESeq [68], incorporate some information across genes. Still, the issue of overfitting and retrieving too many statistical relevant features persists.

2.4.1.2 Trees

Although often used for classification or regression tasks, tree-based approaches perform some inherent feature selection. Decision trees are built upon nodes. The topmost node, which is called the root note, splits the data into two nodes according to the feature which provides the best split [69]. Each node is split again until no further splits are possible or until a pre-defined depth is reached. Although different splitting criteria are possible, the concept of impurity is most prevalent [69]. Impurity describes the degree of uncertainty within a node. In classification tasks, the Gini index is commonly applied. Here, 0 indicates a perfect split while 0.5 denotes maximum impurity. For regression tasks, other impurity measures like mean squared error (MSE) are applied. However, single decision trees are not capable of reflecting the inherent complexity of high-dimensional omics data [69]. One possible extension includes random forest (RF) models, which assemble many different trees, each casting a unit vote [70]. The final prediction is formed through a majority or average vote in classification and regression models, respectively. Even without pruning, i.e., the removal of branches to reduce the complexity of a tree, Breiman et al. found that RFs do not overfit due to the Law of Large Numbers [70]. Practically, single decisions trees are rarely used nowadays, hence further concepts are described in the context of RFs.

After the RF is build, the importance of features for classification or prediction can be estimated. Again, impurity can be used to calculate the contribution of each feature in each node in each tree where it was split [71]. Alternatively, feature importance can be estimated through permutation importance (PI). Here, the values of a feature are randomly permuted in unseen (out-of-bag) data to then determine the model's performance. The precise metric depends on whether the task involves classification (e.g., accuracy) or regression (e.g., MSE). A drop in performance indicates that the feature is relevant for prediction. Both strategies allow for an intuitive ranking of all features according to their importance to the model. However, the linear ranking of feature importance metrics might be misleading for features which are only jointly predictive [72]. Also, both, PI and impurity-based importance (IBI), were found biased in the prevalence of correlating features [73, 74]. One common effect is that not only highly correlated features but also true replicates will receive varying feature importance scores (shown for PI, [75]). This can be attributed to the fact that RFs were initially designed for classification and regression,

but not primarily for interpretation. For interpretation purposes, all relevant features, even highly correlated ones, should be found [64, 75].

The Boruta algorithm can be viewed as an extension of the RF model, aiming to identify all relevant features [76]. Briefly, for each feature, a shadow-variable is created, whose values are obtained through shuffling. The extended model's performance is then evaluated through a RF model and compared against the performance of the original model. Each feature is then marked as *unimportant*, *tentative* or *confirmed* based on Z-score statistics. This procedure is repeated for a pre-defined number of iterations or until all features are either classified as *unimportant* or *confirmed*. Boruta yielded promising results when evaluated on its consistency to select features from omics data of a single breast cancer cohort, available by means of two different technologies [77]. Consistency in this study was defined by the capability of Boruta to mark a consistent set of features as important among all the correlated features present in the omics data. However, it remained unclear to which degree the feature selection was complete or to which extent pairs of highly correlated features were consistently selected.

2.4.2 Feature Extraction

Feature extraction methods generate new features from the original feature space through transformations [78], typically in an unsupervised way. Compared to feature selection, the retrieved features are usually considered less explainable as the link to the original feature space is often obscured. Essentially, the data in the lower dimensional space should preserve certain properties of the original features [79]. Examples of these properties include the pairwise distance such that similar samples in the original space remain similar in the lower dimensional space. Others involve the preservation of local and global structures. Local structures focus on the properties of direct neighbor data points such as co-expressed genes while global structures try to retain broader patterns like gene pathways. How well these properties can be obtained depends on the degree of dimensionality reduction applied and the method used.

Size of Lower Dimensional Space

A survey, evaluating publications in the field of bioinformatics from 1999 to 2006, found that there is usually no systematic approach applied on how to derive the size of the lower dimensional space [80]. The same study found that the original data is often reduced

to less than 10% of the original size. It can be speculated that this number has been further decreased since then as the collected feature space size is ever increasing. While it would facilitate subsequent analyses to reduce the original space to a very small number of features, it would hardly reflect the relationship and geometry of the original dataset. According to the Johnson-Lindenstrauss lemma, the pairwise distance between data points can be nearly preserved when projected into a lower space using linear transformation [81]. However, Chari et al. elaborated that even for a modestly sized dataset of 10,000 data points and a margin of error of 20%, this would require at least 1,842 dimensions [79]. Beyond this theoretical boundary, the size of the feature space needs to suit the precise application.

In the following section, commonly used feature extraction methods are discussed.

2.4.2.1 PCA

The reduction of data through principal component analysis (PCA) is accomplished by finding orthogonal linear combinations of the original dimensions and thereby maximizing the variance [82]. This can be achieved by first computing the covariance matrix of some centered data X. Second, eigendecomposition is performed to derive the eigenvector matrix V and the corresponding eigenvalues. Conceptually, this step can also be achieved through singular value decomposition [83]. The eigenvector with the highest eigenvalue corresponds to first principal components (PC), which is orthogonal and thus uncorrelated to all remaining PCs. The entries of the eigenvector V are the so-called loadings w, with w_{ij} representing the loading of the original feature j on the i^{th} principle component. A principle component PC_i is then defined as the weighted sum of the original features x_j and the derived loadings w_{ij}:

$$PC_i = w_{i1} \cdot x_1 + w_{i2} \cdot x_2 + ... + w_{ip} \cdot x_p$$

where p is the total number of original features.

The loadings w_{ij} can also be utilized to estimate the contribution of an original feature j to PC_i. However, usually many features will contribute to a single PC and likewise, one feature will contribute to many PCs, hampering interpretability.

The so-called score matrix Z represents the data in the lower dimensional space and can be computed by

$$Z = XV$$

If only the first p principle components (and not all components) are utilized, the original data matrix X can be approximated from the score matrix Z using

$$X \approx ZV^T_{(1:p)}$$

How well the reconstructed data approximates the original data will depend on the number of principle components that are chosen. While PCA is applied in many fields, there is no consensus on how many principle components to use, though a variety of methods have been proposed [84]. Often, especially for interpreting and visualizing results, only the first two components are considered. More critically, these two-dimensional representations are also used to draw misleading or inaccurate conclusions about the original data. For example, Elhaik showed for population genetic studies that the findings gained from a PCA plot are affected by the choice of markers, samples and populations [85]. His report concluded that results of PCA are not reproducible for other datasets than the investigated one and are therefore of little use. In terms of retention of properties (pairwise distance, local structure, global structure), PCA will primarily preserve global structures through maximizing the variance in the data. Theoretically, also the distance can be preserved if the data exhibits a high degree of linearity in the original space [82]. However, for features of spatial omics data, which are known to exhibit more complex, non-linear relationships, this assumption will not hold true. It is therefore not surprising that when PCA was applied to MSI data in previous publications, it extracted relatively few informative features compared to other methods [86, 87].

2.4.2.2 t-SNE and UMAP

T-distributed stochastic neighbor embedding (t-SNE) and uniform manifold approximation and projection (UMAP) are both non-linear dimensionality reduction methods, which rather preserve local structures over global structures [88]. Briefly, t-SNE first determines the similarity between data points in the original space, whereas nearby data points are considered similar and distant points are considered dissimilar [89]. The data points are then randomly projected onto a lower dimensional space and their position iteratively

optimized based on their similarities in the original space. This is achieved by minimizing the Kullback-Leibler divergences between the high-dimensional and low-dimensional probability distributions [89]. As a consequence, errors in nearby data points are penalized more than those in distant points, emphasizing the preservation of local structures [89]. UMAP, on the other hand, constructs a fuzzy topological representation of the high-dimensional data, which is then optimized by minimizing the cross-entropy between the topological representations in the high-dimensional and low-dimensional spaces [88]. The topological representations are build by considering the nearest neighbors and their distances to each other, thereby effectively preserving non-linear structures in data. Both methods are not deterministic, as their initialization and applied optimization procedures incorporate some randomness. Similar to PCA, t-SNE and UMAP are often utilized to reduce high-dimensional data to two-dimensional embeddings, particularly in the field of scRNA, as discussed by Chari et al [79]. In that work, they showed that the two-dimensional embeddings from UMAP and t-SNE are not suitable for preserving local or global structures. Even worse, the interpretation of these embedding was found to be contradictory and context-dependent. Chari et al suggested not to rely on generic dimensionality reduction methods, but rather to focus on more targeted analyses and targeted low-dimensional embeddings.

2.4.2.3 Neural Networks

Although neural networks are typically used for classification or regression tasks and not for dimensionality reduction per se, their inherent architecture is designed to perform feature extraction. A neural network (NN) essentially consists of layers, so-called hidden layers, which in turn are interconnected by edges between nodes [90]. Nodes are the components which apply the actual transformations (linear or non-linear ones) to the data by incorporating weights. An example of a non-linear transformation (termed activation function), that is commonly used in NNs is named rectified linear unit (ReLU). It is defined as:

$$f(x) = \max(0, x)$$

The function becomes non-linear as negative values are ”rectified” to zero. The lower dimensional space of a NN is shaped by optimizing a task-specific loss function. The data is typically split into batches for processing. For regression tasks, common loss functions to measure the error between actual and predicted values include mean squared error

(MSE), mean absolute error (MAE) or the Huber loss. The MSE and MAE for a single batch are defined as:

$$\text{MSE} = \frac{1}{b}\sum_{i=1}^{b}(y_i - \hat{y}_i)^2$$

$$\text{MAE} = \frac{1}{b}\sum_{i=1}^{b}|y_i - \hat{y}_i|$$

whereas b denotes the number of samples in a batch and y and $\hat{y}$ denote the actual and predicted values, respectively.

The Huber loss combines the strengths of MSE, i.e., fast convergence, and MAE, i.e., robustness to outliers, by utilizing a parameter (referred to as δ) to determine the transition points between both errors [91]. It is defined as:

$$Huber_\delta = \frac{1}{b}\sum_{i=1}^{b}\begin{cases}\frac{1}{2}(y_i - \hat{y}_i)^2, & \text{if } |y_i - \hat{y}_i| \leq \delta \\ \delta \cdot |y_i - \hat{y}_i| - \frac{1}{2}\delta^2, & \text{otherwise}\end{cases}$$

whereas b denotes the number of samples in a batch and y_i and $\hat{y}$ denote the actual and predicted values, respectively.

While PCA computes the principle components directly, the hidden layers of a NN are updated during an iterative training process. During every iteration (otherwise known as an epoch), an optimization algorithm like adaptive moment estimation adjusts the parameters of a NN, such as the weights, to minimize the loss further. After training, features can be extracted from the encoded representations of the hidden layers in a NN. The layers, their total number, their size and the activation functions are hyperparameters of a NN among others [92], which are configurations that are defined before the actual optimization process. The type of layers can range from fully-connected layers (so-called dense layers), layers optimized for retaining spatial information (so-called convolutional layers) to regularization layers (such as dropout layers) [93]. For processing spatial omics data, convolutional layers are especially intriguing. Convolutional layers usually work on patches, i.e., the input is a three-dimensional matrix, consisting of width, height and

channels. In a spatial omics scenario, the channels represent the molecular features, e.g., gene expression values. A convolutional layer applies multiple filters (so-called kernels) to all the channels of the input [94]. Each filter is limited by the kernel size, defined by a width and a height. As the input patches are typically larger than the defined kernel size, the kernels need to shift along the image coordinates. Therefore, the so-called stride parameter is set, with a stride of one meaning that the kernels shift one pixel at a time while traversing a patch [93]. Commonly used kernel sizes are 3×3 or 5×5 windows. Typically, odd-sized kernels are utilized because of their property of possessing a center pixel, which is used as an anchor point for the kernel while traversing. Even-sized kernels lack a central pixel but may be valuable for identifying asymmetric structures e.g., in biological tissue. The number of filters will determine the number of channels in the output of the convolutional layer. For dimensionality reduction purposes, the number of kernels will be significantly smaller than the number of molecular features. The shape (width and height) of the output is determined by the kernel size, the stride and whether or not padding is applied [93]. Besides considering the spatial context, convolutional layers are also computationally more efficient than dense layers. This arises from the fact that nodes are no longer fully connected and weights are shared in each filter, thereby reducing the number of trainable parameters [95].

NNs with many layers are typically referred to as deep learning methods. In Chapter 5.2, it is evaluated how these architectures might be utilized for combining spatial omics data. Therefore, some common architecture design decisions are summarized in the following: The lower layers (i.e., layers close to the input) in a deep NN typically capture simple patterns (like edges), while higher layers (i.e., layers close to the output) aggregate them to more complex features [96]. This allows for a better abstraction of features and thus enhances the model's ability to better generalize to unknown data. One commonly used deep learning architecture for imaging tasks is named Densely Connected Convolutional Networks (DenseNet) [97]. Different variants with a different total number of layers have been published, e.g., DenseNet121 consists of 121 layers. One building block is the so-called dense block, consisting of l layers, whereas every layer can be a composite function of batch normalization, non-linear ReLU function, pooling or convolution [97]. The distinct aspect of DenseNet compared to other architectures is that every layer is connected to every other layer in a dense block. These dense blocks are stacked and connected through transition layers. Transition layers itself comprise against a composite function as stated above, but

in this case the convolution is applied through 1×1 kernels [97]. Incorporating 1×1 kernels implies that each pixel is considered individually and that the spatial dimensions remain unchanged. Commonly, 1×1 kernels are utilized to reduce the number of channels [98], which is effectively what happens in transition layers in a DenseNet. Additionally, 1×1 kernels significantly reduce the number of trainable parameters, enabling to tackle computational bottlenecks and in turn allow for even deeper networks [98, 99]. Overall, DenseNet and other deep learning approaches significantly impacted various computer vision domains. The proposed architectures are readily available in the most common machine learning libraries (like TensorFlow, Pytorch), enabling to reuse and adjust the models for new tasks. Particularly compelling is that many of these models were trained on large image datasets (such as ImageNet). The weights of these pre-trained models can then be utilized for the training of a related or even different task, referred to as transfer-learning [96]. Along with a reduced time for training new models, the pre-trained models typically acquired already good general purpose features like shapes or textures [100].

One drawback of NNs is that the relationship of the original and the hidden layers is obscured. However, the examination of weights or visualization of hidden layers might reveal some insights into the contribution of the original features. In a highly correlated feature space, contributions are more difficult to derive as many methods are not applicable as they assume feature independence (like Kernel SHAP [101]).

2.4.2.4 Autoencoders

An autoencoder (AE) consists of an encoder and a decoder to compress the high-dimensional data into a lower dimensional space. While the encoder reduces the original feature space, the decoder is typically a mirrored version of it, allowing to reconstruct the original data from the lower dimensional space. Conceptually, the encoder and decoder are most commonly two stacked neural networks [4]. When utilizing NNs, transformations are not limited to linear operations contrary to PCA. However, when only linear transformations are applied, the loadings of PCA can be recovered from the weights of AEs [102]. Thus, AEs can be considered as a generalization of PCA [103]. The last hidden layer of the encoder is typically referred to as latent space. For dimensionality reduction, the latent space size is lower than the input size, imposing a bottleneck which forces

the AE to aggregate the input data. The loss function in an autoencoder optimizes the reconstruction error between the original and reconstructed data. Commonly applied loss functions are MSE or MAE for continuous data, or cross-entropy for binary data, respectively.

AEs encounter the same challenges regarding explainability of the latent space as the hidden layers in a single NN. Whether an AE is able to preserve structures and distances will heavily depend on the underlying architecture. For example, it is obvious that a latent space that is too small will eventually lead to discarding complex local structures. However, given the various types of AEs which have evolved for various applications, there is no general answer to this question. Common AE architectures include:

- Convolutional AEs. Convolutional autoencoders incorporate convolutional layers for compressing and reconstructing the data, particularly imaging data [104].
- Regularized AEs. They integrate regularization techniques such as dropout or sparsity constraints to prevent the model from overfitting to the training data [103].
- Graph-based AEs. Here, relationships and dependencies of an input graph are learned through the process of message passing, allowing to acquire some local neighborhood information [105].
- Variational AEs. These AEs learn the underlying probability distribution of the input data, thus allowing to draw novel samples from it [103]. Due to its ability to generate new samples, it is considered to be a generative model.

In the context of this thesis, the overall aim is to construct a latent space that specifically preserves structures associated with tumor hypoxia.

2.5 Image Co-Registration

Spatial omics data is often complemented by microscopy images, e.g., to derive annotations or to segment specific areas. In this thesis, H&E and fluorescence images are utilized (see Chapter 3), which need to be aligned with the spatial omics data. Therefore, the concept of image co-registration is briefly described in the following.

The overall goal of registration is to find a coordinate transformation which is applied

to the so-called *moving* image in order to fit the so-called *fixed* image [106]. Commonly applied transformations include linear transformations such as translation, rotation, scaling and sheering, and non-linear ones such as elastic transformations. This co-registration process is achieved by iteratively minimizing a cost function using algorithms like gradient descent. The cost function measures the dissimilarity between the transformed *moving* and the *fixed* image. Metrics include sum of squared differences (SSD), gradient-based metrics or mutual information (MI). The choice of metric depends on various factors, including the characteristics of the images and the transformations being applied. For example, sum of squared differences (SSD) is primarily able to capture linear relationships, while mutual information (MI) allows the detection of non-linear associations. Often, the co-registration is carried out in a hierarchical manner, i.e., the images are registered at different levels of details [107]. This can, but not necessarily has to, involve registering the images at multiple resolutions. Typically, multi-resolution registration first aligns global structures followed by finer details (known as "coarse-to-fine" strategy). Many more parameters exist such as the choice of an appropriate sampler for pixel selection or the selection of interpolators for estimating pixel values outside the grid positions [106]. Especially when aligning images from different modalities, co-registration can be considered as a non-trivial task on its own.

3 Materials and Methods

The HNSCC xenograft models used in this project are part of a larger project on the effect of tumor hypoxia (see Koi and Bitto et al [59]). In total, spatial omics data, i.e., mass spectrometry imaging (MSI) and spatial transcriptomics (SPT), of seven different xenograft models were collected. These tumor models originated from HNSCC cell lines, namely FaDu, SAS, UT-SCC-5 (UT5), UT-SCC-8 (UT8), CAL33, UT-SCC-45 (UT45) and SAT, with characteristics summarized in Table 3.1. Small pieces of tumors were transplanted to mice, which were later randomly allocated into one of three groups (one control group, two treatment groups). The control group remained untreated. The mice of the interventional groups received either 10 fractions (fx) of radiotherapy combined with weakly cisplatin (RCTx) and sodium chloride as carrier or 10fx of RCTx + nimorazole (for details see Koi and Bitto et al [59]).

Table 3.1: Cell lines used for biomarker discovery, characteristics according to CelloSaurus [108], female[1]according to Takahashi et al. [109]. Table was first published in Koi and Bitto et al. [59].

Name	Sex	Age	Origin	Anatomical site	HPV status
FaDu	Male	56	South Asia	Hypopharynx	HPV-negative
SAS	Female[1]	69	East Asia	Oral cavity (Tongue)	HPV-negative
UT-SCC-5	Male	58	nd	Oral cavity (Tongue)	HPV-negative
UT-SCC-8	Male	42	nd	Larynx	HPV-negative
CAL33	Male	69	Europe	Oral cavity (Tongue)	HPV-negative
UT-SCC-45	Male	76	nd	Oral cavity (Floor of mouth)	HPV33-positive
SAT	Male	nd	Japan	Oral cavity	HPV-negative

3.1 Data

Upon submitting this thesis, spatial omics experiments for different control and treatment groups had been performed and were available for analysis (Table 3.2). Experiments that had been repeated for the same sample, e.g., to check the consistency of measurements after technical updates of the instruments had been carried out, are not explicitly listed. For the subsequent chapters, analysis will focus on untreated samples only. For treated samples, it was discovered during sample collection that often no or only small hypoxic regions remained intact.

Table 3.2: Total number of spatial omics data collected. Samples treated with 10 fractions (fx) of primary radiochemotherapy (RCTx) with and without nimorazole were not analyzed in this work and are therefore grayed.

	Tumor samples		
Data	Untreated	10fx RCTx + carrier	10fx RCTx + nimorazole
Mass Spectrometry Imaging (MSI)	26	23	25
Spatial Transcriptomics (SPT)	9	5	6
MSI & SPT serials	4	0	0

Considering the results described in Section 2.3.3, the most promising tumor models for biomarker discovery of hypoxia are CAL33 and SAT. Both tumor models are Human Papillomavirus (HPV)-negative and showed no decrease in hypoxia (measured as pimonidazole hypoxic volume) during a combinational treatment of RCTx plus nimorazole or RCTx alone [59]. In contrast, FaDu, SAS, UT5 and UT8 showed a significant decrease in hypoxia when treated with a combinational treatment of RCTx plus nimorazole or RCTx alone after 10 fractions. Therefore, CAL33 and SAT exhibited presumably a more resistant degree of hypoxia compared to the other models, provided that the effect was not predominantly caused by differences in the applied radiation dose per fraction [59]. Arguably, the difference in treatment response of the models could manifest through different genetic imprints, which may be already detectable in untreated samples. Therefore, analyses were carried out independently for the different response models. It is expected that a mixing of the response groups would lead to signal attenuation of the non-responding phenotypes. UT45, being the only HPV-positive tumor model, was excluded for subsequent analysis, given that HPV positivity is associated with a higher radiosensitivity. For UT5 and UT8,

no spatial omics data was collected upon submitting this thesis. Table 3.3 summarizes the individual samples per tumor models which were considered for analysis.

Table 3.3: Tumor models and the available samples considered for this thesis.

	Samples of tumor models (untreated)			
Data	FaDu	SAS	CAL33	SAT
Mass Spectrometry Imaging (MSI)	5	5	5	6
Spatial Transcriptomics (SPT)	3	0	3	3
MSI & SPT serials	1	1	1	1

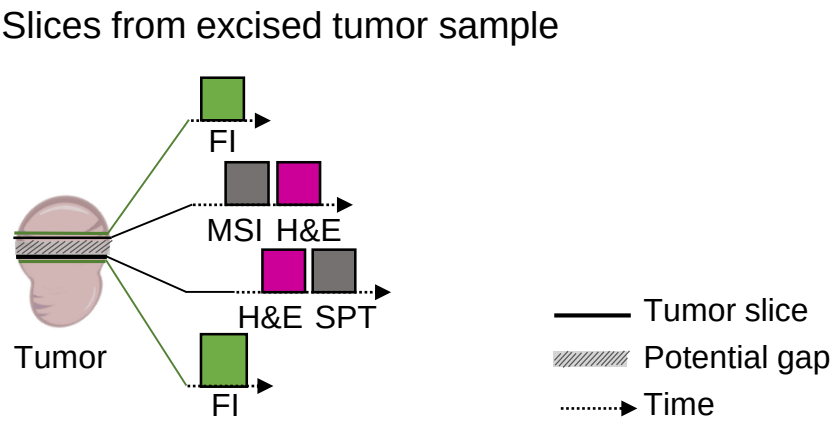


Figure 3.1: Slices cut from one tumor sample. Up to four slices each per tumor sample were cut and processed: One slice was used to generate mass spectrometry imaging (MSI) data and was stained with hematoxylin and eosin (H&E) afterwards. One slice was used to generate spatial transcriptomics (SPT) data and was stained with H&E before the experiment. Two additional slices were stained with pimonidazole, denoted as fluorescence image (FI) in the figure. The thickness of lines visualizes that the thickness of the cut tumor slices differs. The gap illustrates the potential offset caused from cutting the slices at different locations.

The slices on which the spatial omics experiments were performed, were also stained with H&E. Additionally, for every spatial experiment a true serial slice was used for fluorescence staining. Fig. 3.1 gives an overview about the tissue slices and data generated for one sample. While for one sample usually both spatial omics modalities were collected, these tumor slices cannot be considered true neighboring slices (indicated by the gap in Fig. 3.1). This is, because the sample cutting for the spatial omics experiments were initially carried out at different locations (SPT experiments in Heidelberg, MSI experiments in

Dresden). Hence, effects from cutting the tumor at different angles, such as differences in tumor morphology being visible in each slice, were likely to have happened. After discussions with Dr. Denis Schapiro on the amount of structural differences visible in these presumably neighboring slices, my supervisors and I agreed on carrying out some additional experiments. That is, for four samples true serial tumor slices for MSI and SPT data were cut at a partner site in Dresden (used in Chapter 5), referred to as *MSI and SPT serials* in Tables 3.2 and 3.3.

This remaining section describes the data used for biomarker discovery.

3.1.1 (Spatial) Omics Data

The spatial peptide information for this thesis was acquired via MALDI TOF mass spectrometry imaging (MSI), using a α-cyano-4-hydroxycinnamic acid (CHCA) matrix and trypsin during sample preparation. Therefore, tumor sections (1–2μm thickness) were cut from formalin-fixed paraffin-embedded (FFPE) tissue blocks. Measurements were performed on a Rapiflex Tissuetyper (Bruker) in positive reflector mode, whereas each spot has a diameter of 50μm. Additionally, LC-MS/MS experiments were carried out on a Ultimate 3000 UPLC system (Thermo Fisher Scientific) directly connected to an Orbitrap Exploris 480 mass spectrometer. This non-spatial omics bulk experiment was performed for peptide identification in three different untreated samples of CAL33. The masses of the identified peptides were then utilized for matching with m/z values from MSI experiments. The LC-MS/MS experiment, as well as the MSI data of five untreated CAL33 samples have been deposited with the ProteomeXchange Consortium via the PRIDE [110] partner repository with the dataset identifier PXD047820. There, as well as in Bitto et al. [111], also all the MSI and LC-MS/MS protocols are described.

The spatial gene expression data was acquired using the Visium Spatial Gene Expression Solution kit by 10x Genomics [112] from tissue slices cut from FFPE tissue blocks. One expression slide can capture a total of 4992 spots, whereas each spot has a diameter of 55μm, with a center to center distance of 100μm between spots. Depending on the investigated tissue, this will result in an average resolution of 1 to 10 cells per spots.

The following steps were involved in sample preparation (text provided by María José Besso [M.J.B.]):

> "Tissue sections (5μm thickness) were placed on Visium spatial slides following the guidelines from demonstrated protocol CG000408 (10x Genomics). Deparaffinization, H&E staining and decrosslinking were performed following demonstrated protocol CG000409 (10x Genomics). Slide scanning was done using the Olympus VS200 tissue scanner. Construction of Visium gene expression FFPE libraries was conducted following the guidelines described in demonstrated protocol CG000407 (10x Genomics). Library concentration and quality control were performed by Qubit (Invitrogen) and Tapestation (Agilent). Afterwards, libraries were pooled at 10 nM final concentration in 100 microliters and 2x 50 bp paired-end sequencing was performed on the Illumina NovaSeq 6000 S1 according to the manufacturer's protocol."

Apart from the publicly available MSI data via the ProteomeXchange Consortium, all raw spatial omics data is stored under the project ID OE0509 at the Omics IT and Data Management Core Facility of the German Cancer Research Center (DKFZ).

3.1.2 Imaging Data

Tumor slices for which spatial omics data were conducted, were stained with H&E. H&E stains were performed for MSI before and for SPT after the spatial experiment, respectively, on the same tissue slice. H&E images are typically represented in the RGB (Red, Green Blue) color space. However, the primary source of information in H&E stains are the cells' nuclei associated with the blue channel, and the cells' cytoplasm associated with the red channel [42]. The size difference in nuclei also allows to distinguish whether the tissue originated from the mouse as host or from human HNSCC. H&E images were obtained using a microscope at 20× magnification. These images are used in Chapter 5 to combine the two spatial omics modalities as well as to train the MSI data from H&E images using a convolutional neural network (CNN).

Additionally, for every tissue slice that was used in a spatial omics experiment, an immediate neighboring tissue slice (3 μm thickness) was cut to derive hypoxia annotations. These consecutive slices were stained with an anti-pimonidazole polyclonal antibody

(pimonidazole) from Hypoxyprobe™ (Burlington, Massachusetts, USA) as a biochemical marker of chronic hypoxic cells, i.e., the cytoplasm of cells, complemented by 4',6-diamidino-2-phenylindole (DAPI), to label the nuclei of all cells. The resulting image consists of a blue DAPI and a green pimonidazole channel. Fluorescence images were obtained using a microscope at 20× magnification. From every fluorescence image (FI), hypoxia annotations were derived which were used in Chapters 4 and 5 accordingly.

All raw images are stored under the project ID OE0509 at the Omics IT and Data Management Core Facility of DKFZ.

3.2 Software

QuPath (0.4.4) was utilized by one of my colleagues (M.J.B.) to label FIs and to train hypoxia pixel classifiers [113]. The tool was also utilized to export downsampled versions of H&E and FIs.

Image pre-processing was performed using the OpenCV library [114]. Fluorescence images and the corresponding hypoxia binary images were co-registered to the spatial omics data using the ITKElastix (0.17.1) [115, 116] framework for Python. For MSI, also the H&E images were registered via ITKElastix.

For MSI pre-processing, R (4.1.0) and the Cardinal (2.10.0) package was utilized [117]. All other MSI analyses were carried out using Python (3.8.8). For SPT, co-registration of H&E stains with the actual spatial data as well as barcode/UMI counting was carried out using SpaceRanger (1.3.0) by 10x Genomics [118]. SPT data was normalized using the *sctransform* procedure [119] using the Seurat package (4.3.0.1) in R (4.1.0). Further pre-processing of SPT data was executed in Python (3.8.8) using the scanpy (1.9.6) package [120]. The convolutional autoencoder (CAE) approaches were implemented using Tensorflow (2.11.0). The RF models were built using sklearn (1.3.0) [121]. Other packages used include pandas (2.0.3) [122], numpy (1.24.4) [123] and scipy (1.10.1) [124]. The data structure for both, MSI and SPT data, was build upon anndata (0.9.1) [125]. For evaluation of the performance of the Boruta algorithm, originally developed in R [76], the corresponding python implementation boruta_py (0.3) was used [126].

Training of CAEs and CNNs (Section 5.2) were performed via tensorflow-gpu (2.11.0),

cudatoolkit (11.2) and cudnn (8.1.0) using Python (3.8.8). Pre-processing and post-processing of data was performed separately on the CPU. All experiments were run on nodes within the Omics IT and Data Management Core Facility cluster of DKFZ.

The implemented code is presented throughout this thesis, with the overall workflow provided in the Supplementary Material, Chapter 7. The work on MSI is also accessible via GitHub [127] as part of the preprint publication which is available on Research Square [111]. All code is also stored within GitLab (projects: automsi, autospt and hemsiCNN), hosted by the Omics IT and Data Management Core Facility of DKFZ.

3.3 Acknowledgments

This work would not have been possible without all the support provided for wet lab experiments. The cutting of the tumor slices was primarily carried out in the Institute of Pathology, University Hospital Carl Gustav Carus (UKD) in Dresden by Christian Sperling (C.S.). The staining of the fluorescence images, the labeling of hypoxic regions and the corresponding training of a hypoxia pixel classifier was achieved by María José Besso (M.J.B.).

MSI experiments and sample preparation were performed in the Institute of Pathology, University Hospital Carl Gustav Carus (UKD) in Dresden (by C.S). Complementing, LC-MS/MS experiments were carried out in the Proteomics Core Facility of DKFZ in Heidelberg by Dominic Helm and Martin Schneider with protein identification being executed by means of MaxQuant (version 1.6.14.0, [128]).

SPT experiments and corresponding sample preparation were done by M.J.B.

4 Convolutional Autoencoders for Aggregating Spatial Omics Data to Derive Hypoxia-Associated Biomarkers

Spatial omics data shows high potential to reveal novel biological insights to hypoxia as it combines both molecular as well as spatial information of a tumor. Yet, this data imposes several challenges for analysis (summarized in Fig. 4.1).

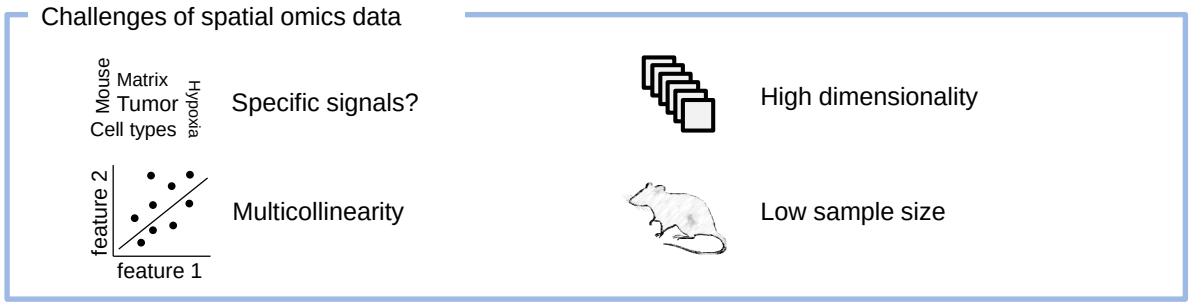


Figure 4.1: Challenges of analyzing spatial omics data

First, the molecular signals associated with tumor hypoxia may only be pronounced weakly compared to other tissue characteristics, e.g., some of the cells originate from mouse as host while others originate from human. Also, some technical signals, such as matrix compounds in case of mass spectrometry imaging (MSI), might mask signals of interest. An aggressive data pre-processing is at risk of discarding promising features,

however the high dimensionality, the high collinearity of features and the low sample size make traditional statistical methods infeasible (see Section 2.4). In the presence of all of these characteristics, dimensionality reduction methods emerge as the most suitable choice for investigating spatial omics data. In context of this thesis, it is imperative that a lower-dimensional space is still preserving hypoxia-associated features.

For comparison, one feature selection method, i.e., a RF model, is compared against a combinational approach using prior convolutional autoencoders for feature extraction. RF models are particularly intriguing, as they can handle all kinds of data, provide an intuitive ranking for feature importance and a proper model setup with only a few hyperparameters, which need to be configured [129]. However, commonly used feature importance metrics are sensitive to highly correlated features [74, 75]. On the contrary, AEs, excel in providing generalized representations of data and thus can handle multicollinearity well if trained properly. Yet, they are typically considered as a black box, characterized by high complexity due to millions of trainable parameters.

In the following, the general workflow is sketched in a data-agnostic way (Section 4.1). This includes the extraction of hypoxia annotations from consecutive tissue slices and the co-registration to spatial omics data. Then, the general idea of implementing an explainable AE architecture is described. Specific results for MSI and spatial transcriptomics (SPT) data are outlined in Sections 4.2 and 4.3, respectively. The chapter concludes with potential limitations and alternative approaches discussed in Section 4.4.

4.1 General Workflow

Spatial omics data of HNSCC xenograft samples were analyzed using three approaches (Fig. 4.2), which were evaluated in terms of their ability to identify relevant features for hypoxia.

Random forest (RF) only: A simple strategy to process spatial omics data could involve the usage of a feature selection method like RF. As decision trees lack spatial awareness, the input data from a spatial omics modality could be either individual pixels or summarized patches, e.g., the mean patch expression value per feature. To minimize the change in parameter setup among the three different approaches, summarized patches are used. These values are used as input together with the hypoxia annotations in order

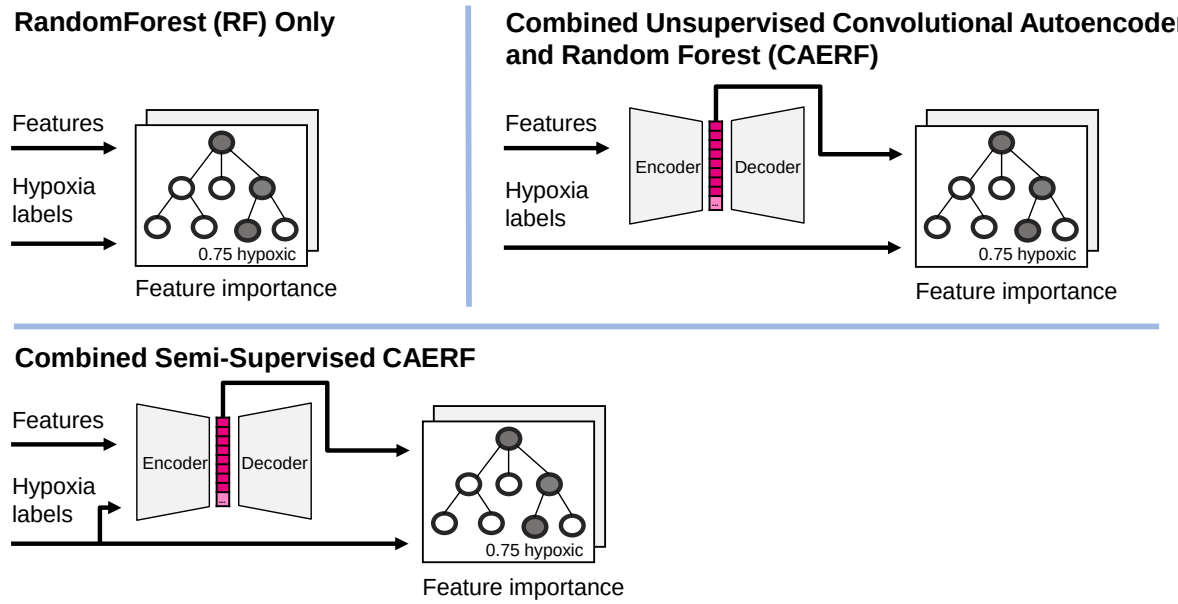


Figure 4.2: Three different approaches to analyze spatial omics data are considered.

to train a RF regression model to predict hypoxia. From the trained model, the features associated with hypoxia are derived according to a mean decrease in impurity feature importance metric.

Combined unsupervised convolutional autoencoder and random forest (CAERF): Instead of directly passing the summarized patches to the RF, an unsupervised convolutional autoencoder (CAE) is compressing the data to lower dimensions first. As a consequence, the subsequent RF can operate on a decreased number of correlations and a reduced feature space.

Combined semi-supervised CAERF: Autoencoders are most commonly utilized in an unsupervised manner. The learned latent representations can be influenced to some extent by the selection of patches and the loss function. Still, without dedicated labels the AE is unaware of the actual features of interest and thus may focus on other, more pronounced features. It may also generalize latent features in an undesirable way. As an extension of the unsupervised CAERF, the proposed combined semi-supervised CAERF approach therefore incorporates the hypoxia annotations while training the convolutional autoencoder (CAE). This allows to assign a higher priority to features linked with hypoxia annotations.

4.1.1 Hypoxia Annotations from Consecutive Tissue Slices

Initially, the consecutive FIs were processed to retrieve binary hypoxic annotations for every pixel (0 = no hypoxia, 1 = hypoxia). After discussions with my supervisors, we agreed to make use of the intensity levels in the fluorescence images and therefore to derive continuous hypoxia values between $[0.0, 1.0]$. This is a more accurate representation of the actual nature of hypoxic cells, which allows to prioritize dense hypoxia clusters over individual hypoxic cells in a patch-wise approach. However, stains are usually noisy, i.e., regions may appear greenish and thus, hypoxic when they are actually just staining artifacts. Ideally, every FI would have been fully annotated, i.e., every cell would be either marked as being hypoxic or not, by an expert. While this might be feasible for one sample, this was not the case for the few dozen images used in this work.

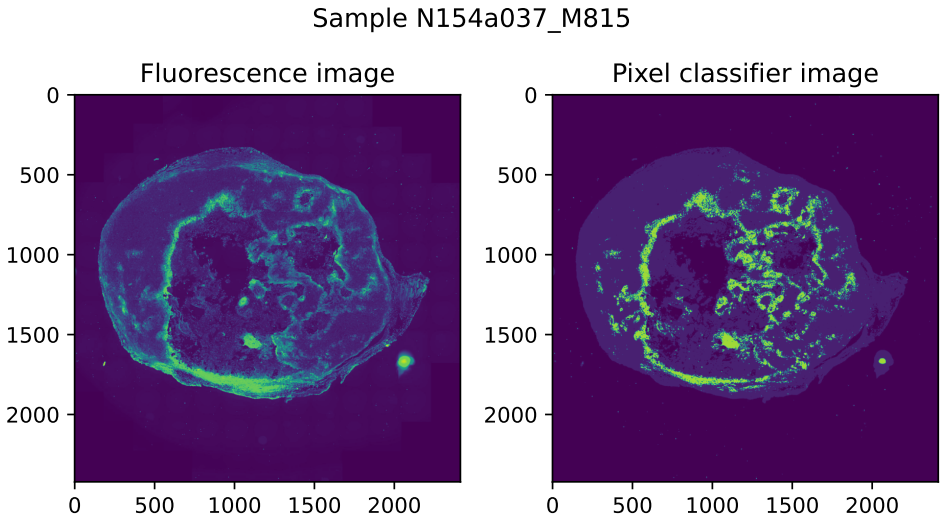


Figure 4.3: Fluorescence image (FI) (left, hypoxia = shades of green, normoxic = shades of blue) and corresponding hypoxia binary image (right, hypoxia = RGB[0,255,0], non hypoxia = hypoxia = RGB[0,0,255]) that was derived with a pixel classifier. The left-hand image (FI) is used to derive the hypoxic value within the range $[0.0, 1.0]$, if a pixel is considered as hypoxic according to the pixel classifier.

As interim solution, small areas of every FI were annotated to train a pixel classifier which aims to discern hypoxic from non-hypoxic cells. The resulting binary image is then used to define which areas are considered as true hypoxic. For every true hypoxic region, the intensity values of the original FI are used to derive the final hypoxic annotation between

$[0.0, 1.0]$. An example of a FI and its corresponding binary image is shown in Fig. 4.3.

In the event the pixel classifier incorrectly assigns a pixel to be hypoxic, dedicated annotations can be used to overrule the pixel classifier. As final step, a median filter is applied to remove isolated hypoxic pixels from the hypoxia annotations and thereby reducing noise.

Co-Registration Of Hypoxia Annotations to Spatial Omics Data

For co-registration, spatial omics data is considered *fixed* while the FI and the corresponding binary image of the pixel classifier are considered *moving*. In the following, capitalized, italic names in brackets denote the corresponding parameter in the ITKElastix framework (see Section Software).

Given that both, spatial omics data and FIs, are inherently different (e.g., in terms of their spatial resolution, scale, contrasts, intensity), the general idea is to rely on the overall shape of a tumor slice for registration. For the spatial omics data, a mean representation of all channels, i.e., the molecular information, is created. The FI and binary image are initially aligned as follows: Images are rotated to a similar position as the fixed image. Then, the background is cropped and the images are approximately downsampled to the size of the fixed image. Following, the fluorescence image is co-registered to the fixed image using an affine transformation model. More precisely, a similarity transform (*SimilarityTransform*) is applied, i.e., the moving image is allowed to be scaled, rotated, translated but not sheered. Even though some deformation might have been occurred during sample preparation of the two tumor slices, I decided to rely on an non-deforming model for co-registration. Registration is performed on 4 different hierarchical resolutions (*MultiResolutionRegistration*, *NumberOfResolutions* = 4). An adaptive stochastic gradient descent (*AdaptiveStochasticGradientDescent*) is used to optimize the mattes mutual information (MI) metric (*AdvancedMattesMutualInformation*), addressing the underlying intensity variations in the images, for a maximum of 2000 iterations. The co-registration was implemented by means of two auxiliary classes *MovingImage* and *ImageRep*, with excerpts shown in Supplementary Material, Code Snippets 7.1 and 7.2. The final transformations are applied to the hypoxia binary image of the pixel classifier. According to this image, the hypoxic spots are identified, and the intensity values of the corresponding FI used as hypoxia annotations. Hypoxia annotations in the range of

$[0.0, 1.0]$ for every pixel position are written to a text file and are added to the spatial omics data during pre-processing.

4.1.2 Explainable Autoencoder Architecture

Autoencoders offer numerous parameters that can be customized, with different architectures accomplishing different aims. Using AEs for dimensionality reduction of spatial omics data needs to consider different attributes than for typical pattern recognition tasks. This includes:

1. For pattern recognition the shape of objects is frequently utilized. However, the shape or border of a tumor does not contain any valuable information in a spatial omics scenario and thus should be disregarded.

2. Genomic and proteomic data is inherently correlated. Thus it is likely that similar features can be collapsed into one latent feature without losing essential information.

3. The lower representations of spatial omics data should express similar characteristics as the original data. Ideally, one latent space feature summarizes multiple original features but is otherwise indistinguishable from them in terms of their expression pattern. In other words, a high abstraction of the original features is unwanted.

4. Neighboring pixels in a spatial omics scenario are likely to share similar molecular patterns that should be taken into account for denoising purposes.

All these characteristics give guidance for the general architecture of the AE. Instead of inputting individual samples, every sample is cut into patches of size $x \times x$ pixels. With that the AE can emphasize local molecular information. Additionally, differences in image sizes are circumvented this way. A sample is therefore padded to a multiple of x, e.g., an image of size 127×121 pixels is padded to 129×129 pixels, if $x = 3$. For the creation and maintenance of patches for some given spatial omics data, auxiliary classes were created, with implementation shown in Supplementary Material, Code Snippet 7.3.

The first two points in the enumeration indicate that a small patch size as well as small kernel sizes might be favorable to achieve the described representations. The second point also suggests that linear transformations alone might not be sufficient to reduce a high-dimensional space significantly in size. Moreover, a low number of hidden layers may

help to depict representations similar to the original feature space. A deep network, i.e., a high number of hidden layers, is likely to enforce a higher abstraction level. The last point in the enumeration implies to use convolutional layers in order to incorporate the spatial context while learning the latent features.

4.1.2.1 Proposed Autoencoder Architecture

Taken into account all of the data characteristics described in the previous section, Fig. 4.4 sketches the implemented AE architecture. In the following, italic names denote the corresponding parameters in tensorflow (see Section Software).

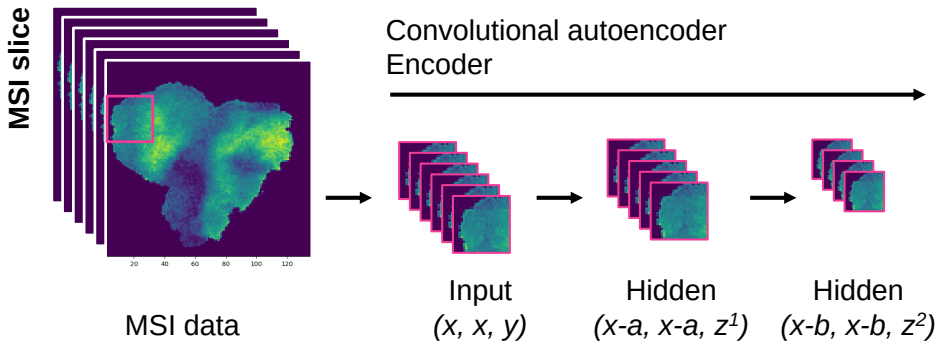


Figure 4.4: Encoder of the explainable AE architecture which uses two convolutional layers with different kernel sizes. Here, x denotes the input patch size, y the original feature space. Variables a and b are dependent on the kernels' stride and the padding, z^1 denotes the feature space in the first hidden layer and z^2 the final latent feature space, whereas $z^1 << y$ and $z^2 << z^1$. This figure was first published in Bitto et al. [111] and was slightly adapted for this thesis.

The first hidden layers consist of a convolutional layer (*Conv2D*) without padding (*padding="valid"*) being applied, followed by a *BatchNormalization* layer and a *ReLU* activation function. This configuration reduces the feature space y to z^1. The bottleneck configuration will enforce to encapsulate similar original features into the same hidden space feature(s). The second hidden layer adds another convolutional layer without padding (*padding="valid"*), followed by a *BatchNormalization* layer and a *ReLU* activation function. This reduces the feature space to the final latent feature space z^2. Depending on the kernel size and the stride, the original patch size is reduced (denoted by the first layer as a and in the second layer as b). For example, a kernel size of 1×1 (*kernel_size*

= 1, *strides* = 1) will not affect x and will effectively ignore neighboring pixels. A kernel size of 2×2 (*kernel_size* = 2, *strides* = 1) will reduce x by one and allows to incorporate also neighboring pixels. The decoder is built symmetrically to the layers of the encoder. While an even kernel size is generally not advisable (see Section 2.4.2.4), the proposed architecture leaded to the most consistent latent representations. The implementation of the CAE is illustrated in Supplementary Material, Code Snippets 7.4.

To approximate the ideal patch size x and kernel size, the visual representation of different configurations are compared. The following base configuration is considered: x = 3, y = 18,735, a = 0, z^1 = 1024, b = 1, z^2 = 64.

4.1.2.2 Effect of Patch Size and Kernel Size

For visualization of the latent space, all patches of an individual sample are encoded using a trained AE. Then the encoded patches of one sample are reassembled to an image, whereas every latent feature can be seen as different image channel.

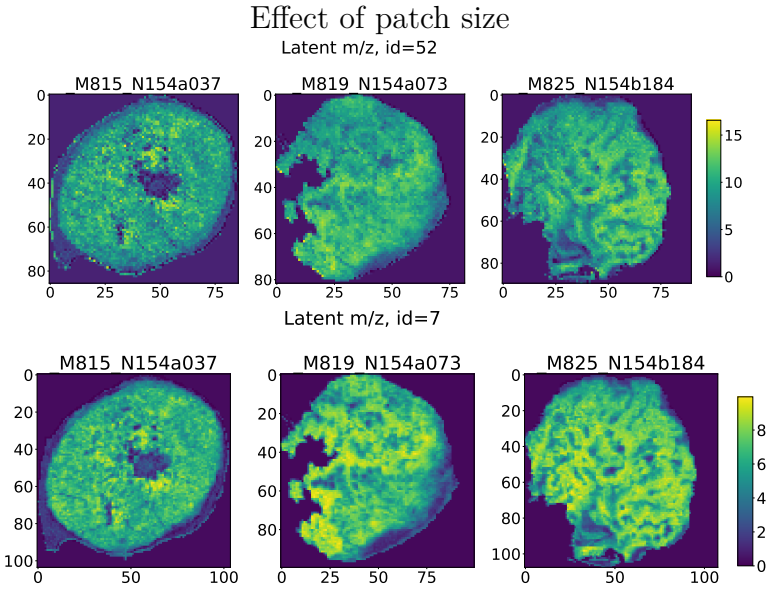


Figure 4.5: Effect of increasing patch size of AE input from $x = 3$ (top) to $x = 5$ (bottom) on similar latent features from two different runs with all other parameters as described.

Fig. 4.5 shows the effect of increasing x from 3 to 5 while all other parameters remain unchanged as described before. Although the visualized latent space features exhibit a high degree of similarity, the latent features are smoother if x is increased. Also, it seems

to lead to the exaggeration of certain signals. While some degree of smoothness may be desirable, it also bears the risk to lose fine-grained details like hypoxic areas.

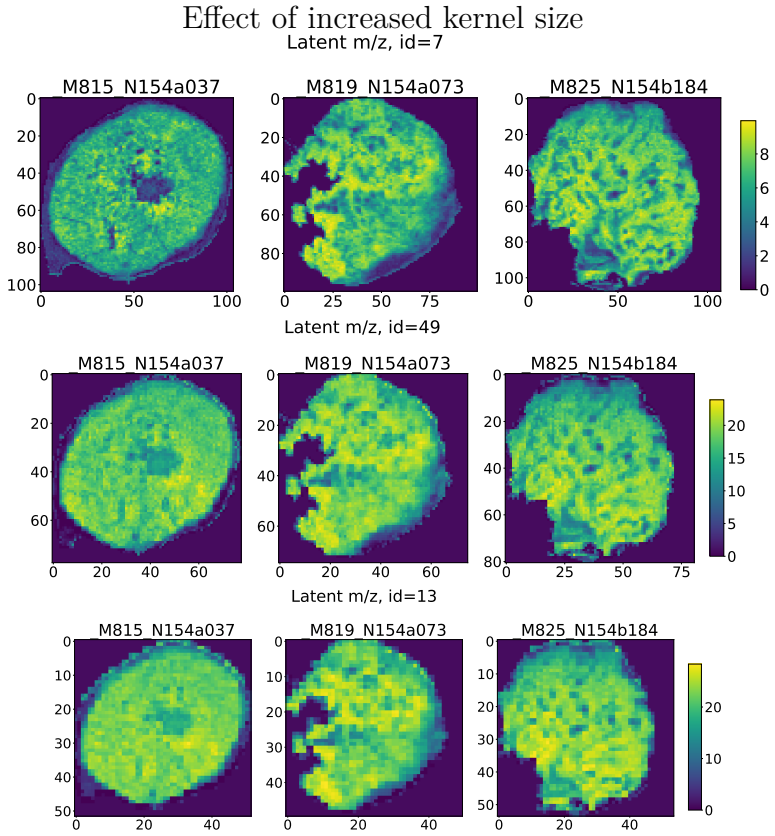


Figure 4.6: Effect of increasing kernel size of second autoencoder (AE) hidden layer from 2×2 (top) to 3×3 (middle) and 4×4 (bottom) on similar latent features from different runs with $x = 5$ and all other parameters as described.

Increasing the kernel size in the second hidden layer from 2×2 (b = 1) to 3×3 (b = 2) and to 4×4 (b = 3) resulted in visible tiles and less details (Fig. 4.6). On the contrary, decreasing the kernel size from 2×2 to 1×1 (b = 0) often yielded to more grainy representations (Fig. 4.7), as no surrounding pixels are incorporated.

4.1.2.3 Effect of Latent Space Size

The final configuration for the hidden feature space size z^1 and z^2, depends on the actual spatial omics data under investigation, e.g., the original feature space size y or the number of noisy features. Usually, the final error, i.e., the reconstruction error in AEs, will serve as an indicator of whether one model performs better than another. However, there might

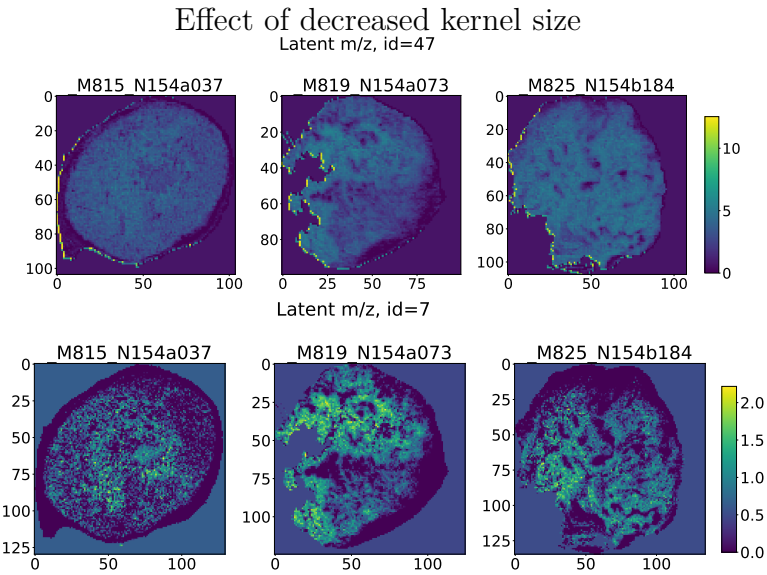


Figure 4.7: Effect of decreasing kernel size of second autoencoder (AE) hidden layer from 2×2 (top) to 1×1 (bottom) on similar latent features from different runs with $x = 5$ and all other parameters as described.

exist configurations with a comparable reconstruction error but differences in how well specific features, like hypoxia-associated features, can be represented. To approach the ideal latent space size, R^2 adjusted from the final RF models to predict tumor hypoxia is evaluated. A high score of R^2 adjusted would indicate that the latent space retains a considerable amount of features associated with hypoxia and is therefore preferred over models with low R^2 scores. The score is adjusted by the number of latent space features, as it is trivial to recognize that the complete original feature space is expected to contain the highest number of hypoxia-related features. A high-dimensional feature space is undesirable as it is prone to produce entangled representations, resulting in multiple latent features being associated with hypoxia. The precise numbers for all parameters and for R^2 are covered in Section 4.2 and 4.3.

4.1.2.4 Recovery Method

The properties of the proposed architecture, i.e., it is relatively shallow and aims to create representations similar to the original features, can be exploited to make the training process of the CAERF to some extent explainable. Given a trained CAERF, the following recovery method is proposed to estimate which original feature contributed to a latent

feature (see also Fig. 4.8):

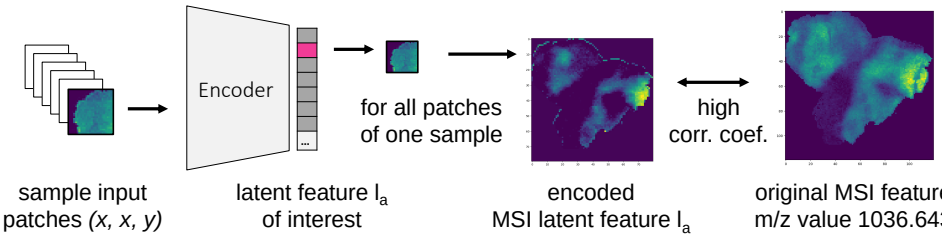


Figure 4.8: Recovery method to identify which original features contributed to a latent feature of interest, exemplified using mass spectrometry imaging (MSI) data. Only one original feature at a time is considered by zero-filling all others. The adjusted patches of one sample are encoded. Then the encoded patches are reassembled to a image for a latent feature of interest which is then compared against its original image. The Spearman correlation coefficient is calculated to derive which original features contributed to a latent feature of interest. This figure was first published in Bitto et al. [111].

For all patches of a given sample, one original feature remains unchanged while all other features are zero-filled. The modified patches are encoded and used to construct a latent representation, i.e., a 2D image, for a latent feature of interest. The original feature representation is downsampled to the size of the latent space representation and subsequent comparison is conducted by calculating the Spearman correlation coefficient. Original features that contributed to a given latent features are expected to show a high correlation coefficient. Likewise, feature which are not associated with a latent feature are expected to show a low correlation coefficient. This procedure is repeated for all original features. A cutoff value (e.g., 0.95) is applied to define the final set of original features that contributed most to a latent feature, such that noisy associations are ideally reduced. Only associations with a p-value < 0.05 are considered statistically significant. As every original feature is considered independently of all other features, highly correlated features are expected to exhibit comparable correlation coefficients for one latent feature. A complete example an how to apply the proposed recovery method is shown in Supplementary Material, Code Snippet 7.5. Auxiliary classes for loading the weights of a trained autoencoder model are presented in Code Snippet 7.6.

4.1.3 Autoencoder Training

To train the unsupervised CAERF, an adjusted MAE loss function is optimized:

$$abs\_diff = |x - \hat{x}|$$

$$mae\_adj = \frac{1}{batches} \sum_{k=1}^{batches} \frac{1}{patches} \sum_{j=1}^{patches} \sum_{i=1}^{features} abs\_diff_{k,j,j,i}$$

where x and $\hat{x}$ denote the actual and predicted intensity values respectively. Here, the error for all intensities in all m/z values is summed up instead of taking the mean. As a consequence, the AE cannot solely favor highly pronounced features but also need to consider low intensity features for achieving a minimal loss. This contributes to preserve supposedly hypoxia-associated features.

For the semi-supervised CAERF, an additional supervised error is calculated as follows:

$$supervised\_error = \frac{1}{pixels} \sum_{p=1}^{pixels} \sum_{i=1}^{features} abs\_diff\_selected_{p,i}$$

whereas the *supervised_error* is only calculated for pixels with a certain degree of hypoxia (cutoff values are defined in Section 4.2 and 4.3, respectively). This allows to weight the error of features with pronounced signals in hypoxic areas. The AE in the semi-supervised mode will then minimize the sum of *mae_adj* and *supervised_error*. The calculation of adjusted mean absolute reconstruction error and the semi-supervised error is shown in methods *CustomLoss.reconstruction_error()* and *CustomLoss.supervised_error()* in the Supplementary Material, Code Snippet 7.7.

The AE is trained with the adaptive moment estimation (Adam) optimization algorithm with a learning rate of $1e-4$ and a batch size of 64. A total of 25 epochs in the unsupervised and 50 epochs in the semi-supervised mode leaded to rapid convergence. In the semi-supervised mode, additionally an early stopping criteria of 30 consecutive epochs is defined to prevent overfitting when no more improvement was achieved on the validation data. An example on how to train the proposed CAE architecture can be found in the Supplementary Material, Code Snippets 7.8 and 7.9.

4.2 Mass Spectrometry Imaging

In this Section, it is investigated if MSI allows to detect signals associated with tumor hypoxia, which can be linked to certain proteins or peptides. MSI is a powerful technology to detect thousands of molecules without the need for target-specific labeling. Yet, there are two major challenges when working with MSI data on top of the general challenges of spatial omics data as summarized in Fig. 4.9.

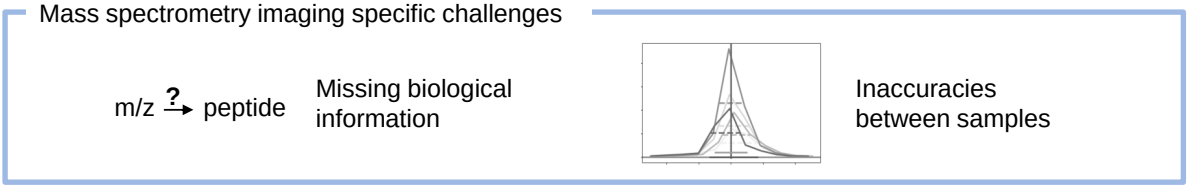


Figure 4.9: Specific challenges of analyzing mass spectrometry imaging data.

First, the identification of proteins and peptides is challenging, as MS typically only outputs m/z values [130]. As a result, the tedious task of peptide identification is often circumvented by only reporting m/z values [36], which is of little use in the context of biomarker discovery. The MSI data in this work is complemented by more precises masses of LC-MS/MS to derive actual peptide candidates associated with tumor hypoxia. Therefore m/z values are transformed to MSI masses that are then linked to the masses of tandem MS. The more MSI m/z values can be assigned to one LC-MS/MS peptide, the higher the probability of an actual match. This is because one individual MS mass might not be indicative for identifying a protein, given that the true mass is obscured by mass inaccuracies.

Second, when analyzing multiple MSI samples, there is no common set of features, i.e., m/z values, due to the nature of mass inconsistencies. This is typically overcome be aligning the m/z values either by a known set of m/z references or through clustering methods [131]. Depending on the type of correction strategy used, this will introduce additional ambiguities in the process, impeding protein identification further. Acknowledging these challenges, I propose to stick close to the raw m/z values to allow peptide identification through subsequent LC-MS/MS analysis.

The precise handling of these challenges are covered in Section 4.2.1. Then, Section 4.2.2

describes how the three approaches (as outlined in the beginning of this chapter) were used to derive m/z value candidates associated with tumor hypoxia.

Of note, the work on MSI was published in part as preprint, available via Research Square [111].

4.2.1 Methods

The sample preparation and the protocols of MSI and LC-MS/MS were not performed by myself and are described in the PRIDE [110] partner repository with the dataset identifier PXD047820. Therefore, this section is confined to the pre-processing of MSI data and the mapping of MSI m/z values to LC-MS/MS masses.

4.2.1.1 Pre-Processing of MSI data

All MSI samples were processed independent of one another and thus differ in their precise 53,400 raw m/z values. The uniform distance between all m/z values was 0.0487. For retrieving a common set of m/z values the following steps were applied:

1. Raw m/z values among all samples are sorted to derive 53,400 mean m/z values, depicted in Fig. 4.10 as columns in row *raw m/z values*.

2. Peak picking on the mean spectra is applied for each sample individually with a signal-to-noise ratio (SNR) of 6 using the Cardinal package to derive sample specific peaks. Exemplary peaks are depicted in Fig. 4.10 in row *mapped peaks*.

3. Sample-specific peaks are assigned to their respective closest mean m/z value, determined through a binary search. As a consequence, peaks of different samples but similar values are assigned to the same mean m/z value (e.g., 601.3146, 601.3244 in Fig. 4.10).

4. These groups of m/z values are then used to derive the final reference peaks using the group's mean, illustrated in Fig. 4.10 as *Peak references (mean)*.

5. (Optional) The number of reference peaks can be reduced by only considering groups with at least y assignments, whereas y can be defined by a parameter. It might be desirable to restrict the total number of reference peaks, as the number is likely to

increase with the number of samples being processed.

6. Before binning each sample to the 18,735 reference peaks, their spectra is normalized by the total ion current (TIC) using the Cardinal package.

7. Intensity values among samples are scaled by a global factor per m/z value to compensate for signal differences between measurements as described by Veselkov et al. [131].

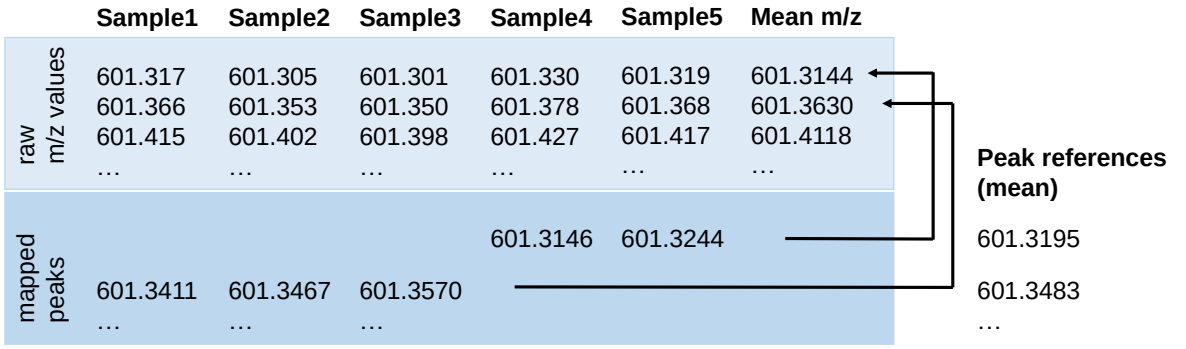


Figure 4.10: Deriving peak references from multiple samples. First, mass-to-charge ratio (m/z) values of all samples are sorted to define a set of mean m/z values. Second, peak picking on the mean spectra per sample is applied (not shown) and peaks mapped to the mean m/z values. M/z values from different samples but similar masses (e.g., 601.3146, 601.3244) will map to the same mean m/z value. These groups are utilized to derive peak references by taking their mean m/z value. (e.g., 601.3195 from peaks 601.3146 and 601.3244). This figure was first published in Bitto et al. [111].

Steps 2–4 were implemented independently from currently existing packages and are provided in R Code Snippet 4.1, whereas *mean_peaks* denote the sample-specific peaks and *raw_mean* the 53,400 mean m/z values, respectively. This pre-processing procedure will retain potential mass shifts, if m/z values of different samples are too different to be grouped together (showcased in Fig. 4.11). This reduces the effect of creating artificial features which deviate significantly from the original m/z values. Likewise, keeping potential mass shifts increases the probability of matching MSI m/z values to LC-MS/MS masses and acknowledges the fact that the exact mass is obscured. Processing these mass shifts using the proposed AE architecture poses no additional challenges, since they can be easily summarized.

Code Snippet 4.1: Extract of proposed preprocessing of m/z values

```
preprocess_mzs <- function(mean_peaks, raw_mean_mz, y, tol) {
  mean_peak_mzs <- lapply(mean_peaks, mz)
  all_mean_peak_mzs <- sort(unlist(mean_peak_mzs))

  # step 2
  # group all_mean_peak_mzs to raw_mean_mz
  idx <- matter::bsearch(all_mean_peak_mzs, raw_mean_mz, tol=tol/2, tol.ref = "none")

  # step 3
  mz_bin <- split(all_mean_peak_mzs, idx)
  mz_groups <- unlist(lapply(mz_bin, length))

  # step 4 (optional, effective if y > 1)
  mz_final <- mz_bin[mz_groups >= y]
  # step 3 proceeding
  mz_final_mean <- lapply(mz_final, mean)
  mz_final_mean <- unlist(mz_final_mean, use.names = FALSE)
  return(mz_final_mean)
}
```

From the 18,735 m/z values, 56 were found to be true replicates, i.e., their intensity values of all pixels were identical to another m/z value. Additionally, around 320 more m/z values were considered as replicates, given that their intensity values were very similar (pearson correlation coefficient > 0.975) to up to two direct neighboring m/z values.

4.2.1.2 Autoencoder Setup

The following configuration (see Section 4.1.2.1) was utilized: x = 3, y = 18,735, a = 0, $z^1 = 1024$, b = 1, $z^2 = 64$–256.

4.2.1.3 Sample Size

MSI data of five samples of untreated tumor model CAL33 was pre-processed and split into patches of size 3×3 pixels. Likewise, the co-registered hypoxia annotations from consecutive tumor slices were cut into patches of size 3×3 pixels. All data was scaled to a range between [0, 1]. To generate more data, overlapping patches with a step size of 2 were generated, from which patches containing no MSI data but only background were removed. The CAE (unsupervised and semi-supervised) was then trained on 8,649 patches from three samples. For validation 2,158 patches of two other samples were utilized.

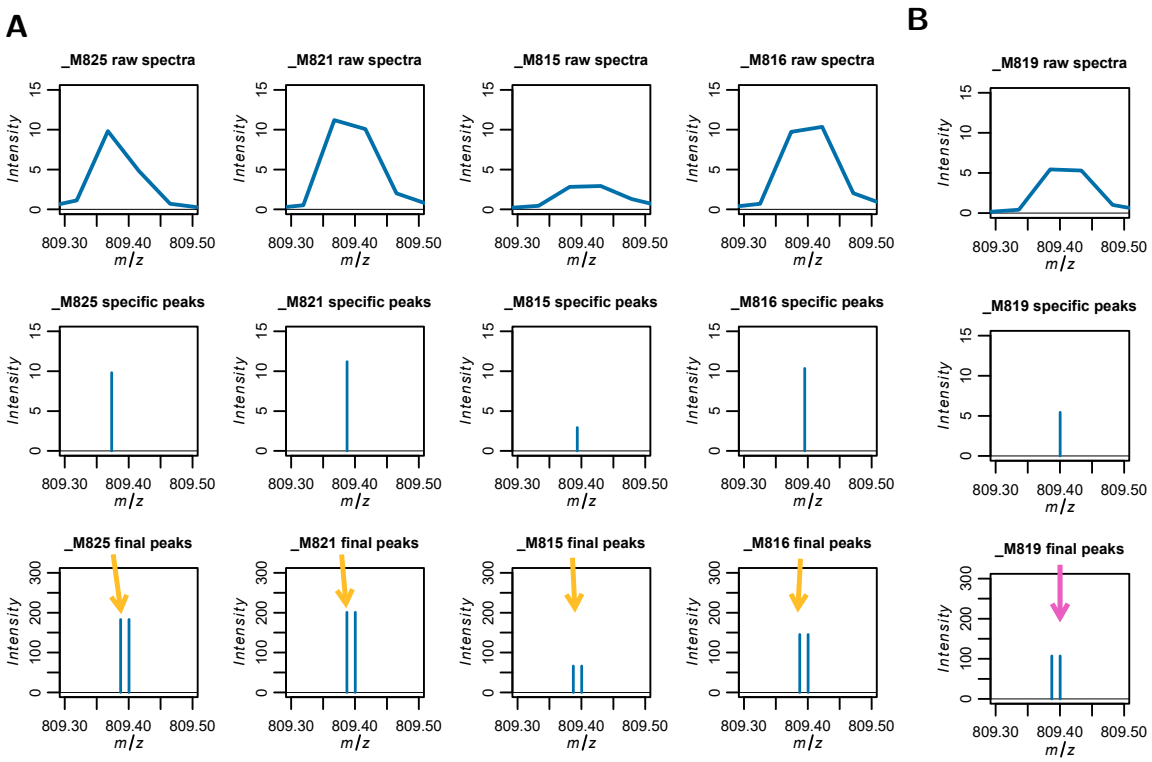


Figure 4.11: Peak picking on multiple samples. Shown is the mean intensity of all pixels in the range of mass-to-charge ratio (m/z) values 809.30 to 809.50 per sample. Row #1 shows the mean raw spectra per sample. Row #2 shows the peaks derived from the mean spectra of a given sample. Row #3 illustrates the final TIC-normalized peaks for all samples. (A) For Samples M825, M821, M815 and M816, their sample-specific peaks match best with mean m/z value 809.387 (yellow arrow). (B) For Sample M819, its sample specific peaks match best with mean m/z value 809.400 (pink arrow). The two peaks around 809.387 and 809.400 are likely denoting the same mass, with inaccuracies caused due to mass shifts. Given that the true mass is unknown, both peaks are kept for further analysis. This figure was first published in Bitto et al. [111] with naming of the samples being adapted for this thesis for consistency.

The three samples used for training the AE, were also utilized for evaluating the performance on the subsequent RF regression models. For the other two samples, the number of hypoxic regions was too low and samples were therefore dismissed. Again, overlapping patches with a step size of 1 were generated. The number of non-hypoxic patches were downsampled to enable the RF to distinguish hypoxic and normoxic features. Therefore, all patches with a mean hypoxia expression of at least 0.12 were chosen. Considering that

this will include many patches with a comparable low hypoxia signal, only 50% additional non-hypoxic patches were added. The RF regression model was then run on 6,409 patches using 10-fold cross validation with a test size ratio of 33%.

4.2.1.4 Random Forest Setup

In the following italic names in brackets denote the corresponding parameter in the sklearn library.

The RF regression models were built upon 1,000 trees (*n_estimators = 1000*) using the MSE as splitting criteria (*criterion = "squared_error"*). The number of features randomly selected at each split was set to the square root of the total number of features (*max_features = "sqrt"*). The patches of MSI and hypoxia annotations were reduced to a mean patch expression value before inputting the values into the RF. Features correspond either to individual m/z values in the RF only approach or latent features in the CAERF approaches.

After training, IBI, i.e., mean squared error, was used to rank important features of hypoxia. The rankings of the test samples according to cross validation outcomes were used to define the feature set associated with hypoxia. A complete example on how to derive the feature importance metrics for all approaches is showcased in Supplementary Material, Code Snippet 7.10, with auxiliary classes shown in Code Snippet 7.11.

4.2.1.5 Mapping of MSI M/Z Values to LC-MS/MS masses

For matching MSI m/z values to masses from LC-MS/MS, m/z values which were associated with hypoxia were converted to masses using the formula:

$$msi\_mass = msi\_mz\_value * 1 - 1$$

where the constant 1 corresponds to the ion charge. This formula is built on the assumption that the majority of ions produced in MALDI MSI are single charged [36].

The peptide information and the corresponding masses of LC-MS/MS were derived using MaxQuant. In total 28,487 peptides and 3,160 proteins were identified based on an FDR cutoff of 0.01 on both, peptide and protein level. Modified proteins, indicated by a "C", were excluded for further analysis.

When mapping MSI masses to LC-MS/MS masses, one has to consider that the observed mass in MS experiments is usually not equal to the actual mass. In LC-MS/MS, for every identified peptide an mass error of the observed and actual mass is calculated. For example, the highest error in the tandem MS experiment was observed for protein *ATP-dependent 6-phosphofructokinase, platelet type*, with an observed mass of 1818.86781 and a mass error of -4.4863 parts per million (ppm). The formula [132]

$$\text{Mass Error [ppm]} = \frac{(\text{Observed mass} - \text{Actual mass})}{\text{Actual mass}} \times 10^6$$

can be rearranged to calculate the actual (theoretical) mass of the protein using

$$\text{Actual Mass} = \frac{(\text{Observed mass} \times 10^6)}{\text{Mass Error [ppm]} + 10^6}$$

which is approximately 1818.87597. As the errors in the LC-MS/MS experiments are comparable small, they were ignored for the mapping of masses. More challenging is the approximation of mass errors for MSI data. As the MALDI MSI measurements had been performed on a RapiFlex Tissuetyper, designed for high-resolution imaging, the full width at half maximum (FWHM) was used as an approximation for the actual mass range. The FWHM denotes the peak width at half of its maximum intensity, i.e., the actual mass is expected to lie within the FWHM. For multiple samples, the FWHM was then calculated for every peak and sample individually, The standard error was then defined by the FWHM of all samples, and can be considered as maximum error. The *lower half-maximum point*, respectively *larger half-maximum point* is the point left respectively right to a peak where it reaches half of its maximum. Additionally, the distance between two m/z values, which is 0.0487, was defined as minimal technical error. The *lower technical point*, respectively *upper technical point* is then defined as the peak - 0.0487, respectively peak + 0.0487. Then, a LC-MS/MS mass was mapped with an MSI mass, if its mass was within the MSI mass range, i.e.,

$$\text{MSI mass min} <= \text{MS/MS mass} <= \text{MSI mass max}$$

whereas

$$\text{MSI mass min} = \max\left(\min_{i=1}^{n} \text{lower half-maximum points}_{\text{sample}_i}, \text{lower technical point}\right)$$

$$\text{MSI mass max} = \min\left(\max_{i=1}^{n} \text{upper half-maximum points}_{\text{sample}_i}, \text{upper technical point}\right)$$

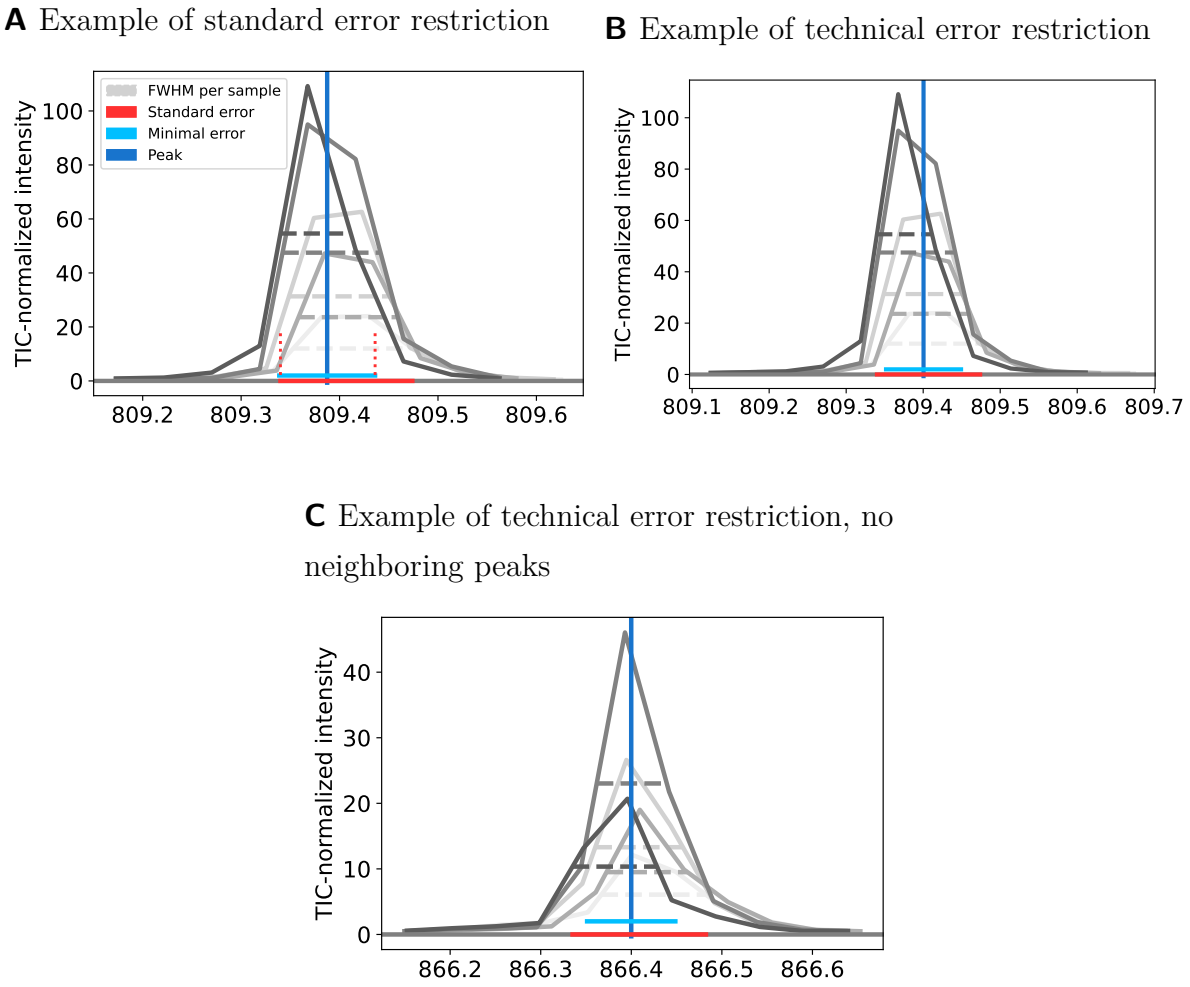


Figure 4.12: Presumed mass ranges for mass spectrometry imaging (MSI) peaks for mapping with liquid chromotography (LC)-MS/MS masses. (A) Standard error (red line), given by the full width at half maximum (FWHM) of all samples, restricts the technical error (blue line) at peak - 0.0487. (B) Technical error of peak is within the range of the standard error and is not further restricted. (C) Peak without (preserved) neighboring peaks that could be linked to mass shifts. Considering the technical error leads to a more conservative range for mapping tandem MS masses than the standard error. This figure was first published in Bitto et al. [111].

and n denotes the number of samples.

Fig. 4.12 illustrates some example peaks and the impact of the defined errors. *MSI mass min* and *MSI mass max* were primarily restricted by the technical error (blue line) to avoid an overly confident mapping of MSI masses to LC-MS/MS masses. However, in

some cases this approach might be still too optimistic. Therefore, the MSI mass range was further restricted by the standard error. Fig. 4.12A and 4.12B show two neighboring peaks, sharing the same standard error given by the minimum lower half-maximum points and the maximum upper half-maximum points (shown in red). In Fig. 4.12A, *MSI mass min* is further restricted by the minimum lower half maximum points (dotted red line), which does not become effective in the neighboring peak (Fig. 4.12B). Using this combined error to limit the MSI range is especially conservative in mapping masses if peaks are not backed up by mass shifts (e.g., Fig. 4.12C).

To reduce the impact of random matches, peptides were only considered as candidates if at least two distinct MSI masses (without potential mass shifts) were assigned. Adding another constraint, the ion images of two mass pairs were expected to correlate with one another (Spearman correlation coefficient > 0.80).

4.2.2 Results

In total five MSI samples of untreated HNSCC model CAL33 were analyzed. For every MSI sample, a consecutive tumor slice was stained with pimonidazole to derive hypoxia annotations. All five MSI samples were pre-processed and a total of 18,735 common peaks per pixel retained. The three approaches were described in the beginning of this Chapter. Figure 4.13 summarizes how the unsupervised CAERF approach differs from the RF only approach.

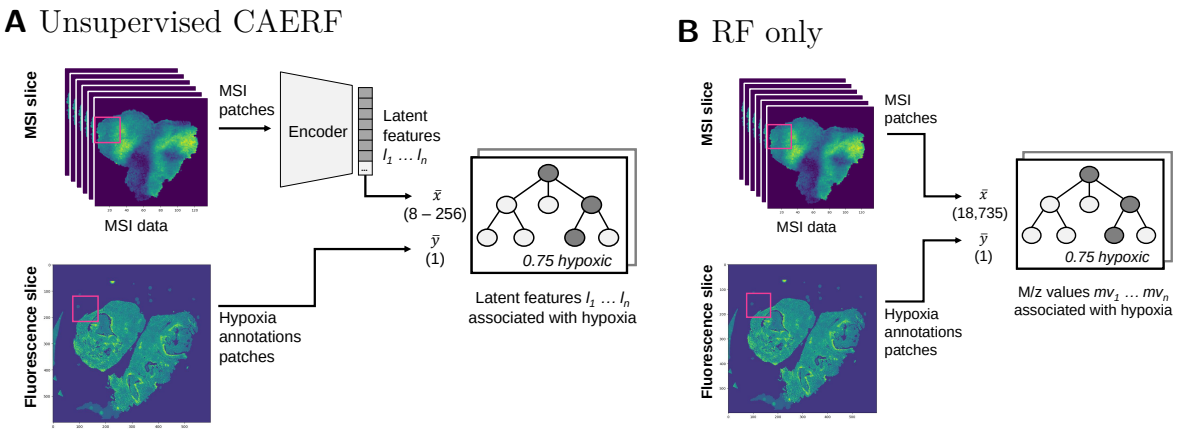


Figure 4.13: Workflow of convolutional autoencoder and random forest (CAERF) and random forest (RF) only approach based on patches. (A) Combined CAERF approach: mass spectrometry imaging (MSI) data was encoded using the trained autoencoder. RF regression models were trained on the encoded data and the reduced hypoxia annotations from consecutive slices. (B) RF only approach: patches of MSI data and hypoxia annotations were reduced to the mean patch values to train a RF model. Numbers denote the length of the input vector for the RF regression model, i.e., 8–256 latent space features or 18,735 original features. This figure was first published in Bitto et al. [111].

As both, AE as well as tree-based methods, involve some degree of randomness, different runs may yield to different results. Therefore, the different approaches were evaluated by qualitative measures of individual runs and quantitative metrics of multiple runs in the following.

4.2.2.1 Qualitative Results of One Exemplary Run Each

Fig. 4.14 shows the hypoxia annotations of the three out of five samples for which hypoxia annotations were available.

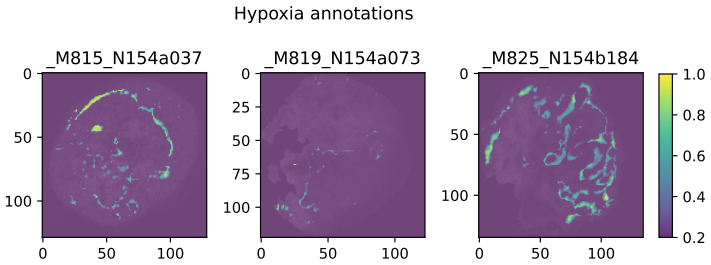


Figure 4.14: Hypoxia annotations of individual mass spectrometry imaging (MSI) samples. Yellow = high degree of hypoxia, dark blue = no hypoxia. This figure was first published in Bitto et al. [111] with naming of the samples being adapted for this thesis for consistency.

Results of Unsupervised Convolutional Autoencoder Run

The CAERF was trained with a latent space size z^2 of 64. Out of the 64 latent features, #56 exhibited the highest RF feature importance for hypoxia. Fig. 4.15 shows a visual representation of latent feature #56 (left) and, for comparison, also a representation for a latent feature with a low feature importance for hypoxia is shown (right).

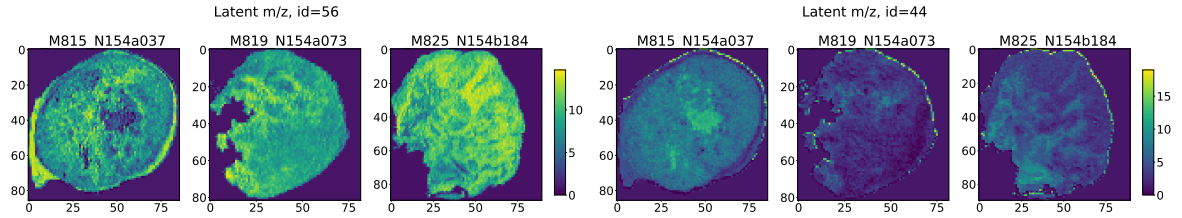


Figure 4.15: Visual representations of encoded mass spectrometry imaging (MSI) samples in latent space. Left: Latent space feature #56 associated with hypoxia according to random forest (RF) feature importance. Right: Latent space feature #44 with low hypoxia association according to RF feature importance. This figure was first published in Bitto et al. [111] with naming of the samples being adapted for this thesis for consistency.

The recovery method described in Section 4.1 was applied to identify which original m/z values contributed to latent feature #56. With a defined Spearman correlation coefficient cutoff value greater than 0.95, 180 m/z values contributed to the hypoxia-associated latent feature #56. Some exemplary m/z values are shown in Fig. 4.16A and 4.16B.

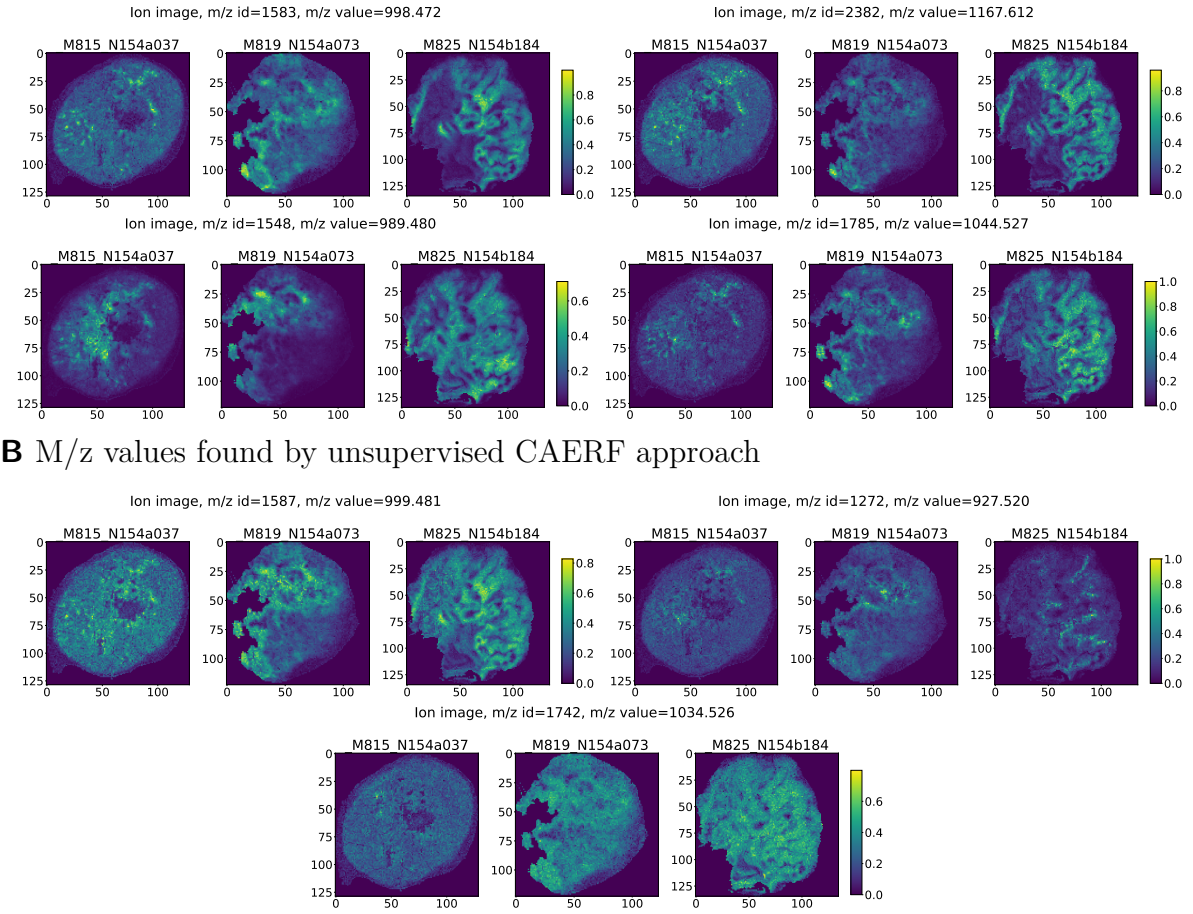


Figure 4.16: Exemplary mass-to-charge ratio (m/z) values associated with hypoxia that (A) were found by both convolutional autoencoder and random forest (CAERF) approaches and the random forest (RF) only approaches or (B) were distinctively found by the unsupervised CAERF approach. This figure was first published in Bitto et al. [111] with naming of the samples being adapted for this thesis for consistency.

Examining the overall value of the features derived by the proposed CAERF approach, the 180 m/z values associated with hypoxia were mapped to the masses of LC-MS/MS experiments. In total 50 peptide candidates were identified for which at least two individual MSI masses could be assigned to LC-MS/MS masses (Table 4.1). Several of those peptides were associated with tumor hypoxia before. For example, enzymes of the glycolytic pathway, like phosphoglycerate kinase 1 (PGK1), pyruvate kinase M (PKM), lactate dehydrogenase A (LDHA) and aldolase A (ALDOA) are known to adopt to low oxygen

supply [43, 133]. Also serine hydroxymethyltransferase (SHMT2) was found to effect cell survival under hypoxic conditions [134]. Other candidates like Annexin A1 (ANXA1) are more generally associated with tumor aggressiveness, but their precise role in HNSCC are still under investigation [135]. Supported by evidence from previously published work, the association of many of the identified peptides candidates to hypoxia seems plausible. The connection of other candidates to tumor hypoxia demands further evaluation.

Table 4.1: 50 peptide candidates found with at least 2 masses from unsupervised convolutional autoencoder and random forest (CAERF) run matched to liquid chromotography (LC)-MS/MS experiment. Only one exemplary mass pair is shown, the complete data is provided in Suppl. Table 1 in Bitto et al. [111].

Protein(s)	Gene name(s)	Mass 1	Mass 2
DNA-dependent protein kinase catalytic subunit	PRKDC	877.466	1337.665
Keratin, type II cytoskeletal 6A;Keratin, type II cytoskeletal 6C;...	KRT6A;KRT6C;KRT6B	877.441	808.387
Annexin A1	ANXA1	808.400	1063.564
Cytochrome b-c1 complex subunit 1, mitochondrial	UQCRC1	808.400	1042.519
Phosphoglycerate kinase 1	PGK1	808.400	1011.519
Elongation factor 1-gamma	EEF1G	809.402	937.455
Keratin, type II cytoskeletal 5	KRT5	809.402	1409.733
Cullin-associated NEDD8-dissociated protein 1	CAND1	988.480	965.469
Serine hydroxymethyltransferase, mitochondrial;...	SHMT2	988.480	854.495
Heat shock cognate 71 kDa protein	HSPA8	988.480	1409.694
RNA-binding protein with serine-rich domain 1	RNPS1	989.471	864.406
Eukaryotic translation initiation factor 3 subunit L	EIF3L	989.471	964.486
Collagen alpha-3(VI) chain	COL6A3	989.471	1036.529
Desmoplakin	DSP	1011.490	944.515
Heterogeneous nuclear ribonucleoprotein R;...	HNRNPR;SYNCRIP	1011.490	926.481
Fatty acid-binding protein, epidermal	FABP5	1042.547	926.520
Prelamin-A/C;Lamin-A/C	LMNA	1042.547	1027.527
Tropomyosin alpha-4 chain	TPM4	1042.547	1259.603
Trifunctional purine biosynthetic protein adenosine-3;...	GART	1042.547	1036.548
Eukaryotic translation initiation factor 4 gamma 1	EIF4G1	1026.520	1410.739
L-lactate dehydrogenase A chain	LDHA	1026.520	1166.637
Programmed cell death protein 6	PDCD6	997.502	1338.656
60S ribosomal protein L18a	RPL18A	1042.519	926.520
Eukaryotic translation initiation factor 3 subunit C;...	EIF3C;EIF3CL	1042.519	1166.637
Keratin, type I cytoskeletal 14	KRT14	1036.529	1166.637
Fructose-bisphosphate aldolase A;Fructose-bisphosphate aldolase	ALDOA	1043.548	939.462
60S ribosomal protein L15;Ribosomal protein L15	RPL15	1166.612	880.449
ATP-dependent RNA helicase A	DHX9	990.459	1074.523
Tubulin alpha-1B chain;Tubulin alpha-4A chain;...	TUBA1B;TUBA1C;TUBA1A;...	1409.733	774.394
Eukaryotic translation initiation factor 3 subunit A	EIF3A	1409.733	816.437

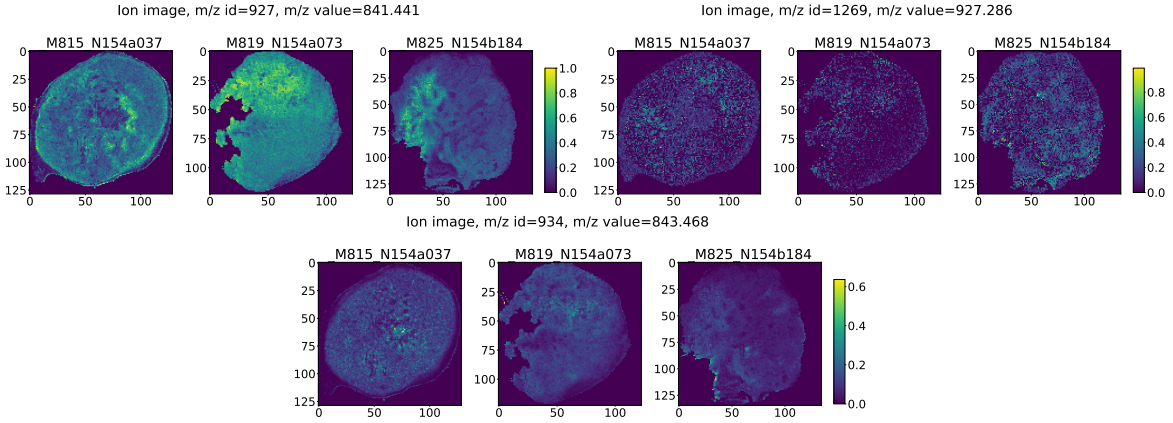
Table 4.1: Continued: 50 peptide candidates found with at least 2 masses from unsupervised convolutional autoencoder and random forest (CAERF) run matched to liquid chromotography (LC)-MS/MS experiment. Only one exemplary mass pair is shown, the complete data is provided in Suppl. Table 1 in Bitto et al. [111].

Protein(s)	Gene name(s)	Mass 1	Mass 2
Elongation factor 2	EEF2	1012.508	879.436
Pyruvate kinase PKM;Pyruvate kinase	PKM	883.451	1167.621
Heterogeneous nuclear ribonucleoproteins A2/B1	HNRPA2B1;HNRNPA2B1	1409.694	1337.665
Aconitate hydratase, mitochondrial	ACO2	1411.719	921.446
Isoleucine–tRNA ligase, cytoplasmic	IARS	1411.719	957.533
T-complex protein 1 subunit theta	CCT8	998.516	1167.580
Bifunctional glutamate/proline–tRNA ligase;...	EPRS	965.469	1063.564
Keratin, type I cytoskeletal 16	KRT16	1337.665	854.495
EH domain-containing protein 4	EHD4	1337.665	937.455
Keratin, type II cytoskeletal 75	KRT75	921.446	1038.511
Myeloperoxidase;Myeloperoxidase;89 kDa myeloperoxidase;...	MPO	921.446	937.455
Transketolase	TKT	921.446	944.515
60 kDa heat shock protein, mitochondrial	HSPD1	854.495	1007.493
Annexin A4;Annexin	ANXA4	856.472	1074.523
Actin, cytoplasmic 1;Actin, cytoplasmic 1, N-terminally processed;...	ACTB;ACTG2;ACTA2;...	944.515	1197.680
26S proteasome non-ATPase regulatory subunit 3	PSMD3	1411.683	957.533
Cyclin-dependent kinase 1	CDK1;CDC2	772.417	1027.527
Transitional endoplasmic reticulum ATPase	VCP	1074.523	1050.523
Heat shock protein beta-1	HSPB1	1074.523	940.463
Activated RNA polymerase II transcriptional coactivator p15	SUB1	1197.680	1259.603

Results of Random Forest Only Run

In the RF only approach, the 18,735 original m/z values were processed. The RF feature importance revealed that m/z value 998.472 was the most indicative for the hypoxia annotations (in 4 out of 10 cross validation runs). For comparison with the CAERF approach, a cutoff based on the highest ranked feature score was defined to retrieve a comparable amount of m/z value candidates. All m/z values that reached at least a fourth of the highest score in a given cross validation run were considered to be associated with hypoxia. This resulted in 156 m/z values, considering all cross validation runs. Some exemplary m/z values are shown in Fig. 4.16A and 4.17A.

A M/z values found by RF only



B M/z values found by semi-supervised CAERF approach

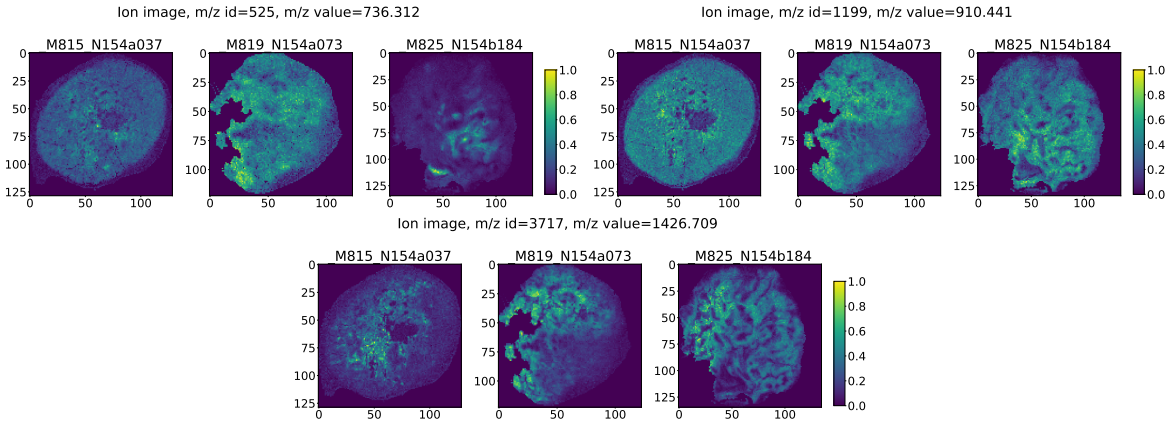


Figure 4.17: Exemplary mass-to-charge ratio (m/z) values associated with hypoxia that (A) were distinctively found by the random forest (RF) only approach or (B) were distinctively found by semi-supervised convolutional autoencoder and random forest (CAERF) approach. This figure was first published in Bitto et al. [111] with naming of the samples being adapted for this thesis for consistency.

Results of Semi-Supervised Convolutional Autoencoder Run

To reduce the number of noisy hypoxia assocations (e.g., m/z value 1034.526 in Fig. 4.16B), the semi-supervised CAERF approach incorporates the hypoxia annotations during training. Given one exemplary semi-supervised CAERF run, the recovery method found 120 m/z values to be associated with hypoxia, whereas 75 of them were already showing up in the unsupervised CAERF approach. Some exemplary m/z values which were distinctively found by the semi-supervised CAERF approach are shown in Fig. 4.17B.

Linking the 120 m/z values to masses from LC-MS/MS resulted in 43 peptide candidates (Suppl. Table S1). Among them, 23 peptides were also observed in the unsupervised CAERF approach (like PGK1 and LDHA). Some promising hypoxia-associated candidates were not retained (PKM, ALDOA) but others appeared. For example, glyceraldehyde-3-phosphate dehydrogenase (GAPDH) was found the be upregulated in various cancer types and appears to be influenced by hypoxia among other mechanisms [136].

Comparison of CAERF and RF Only Approaches: Hypoxia Associated M/Z Values

All approaches shared a total of 42 common m/z values of which 4 are shown in Fig. 4.16A. Consistent to the known limitations of RF when it comes to correlated features, replicate m/z values did not obtain an identical feature importance score, but were usually within the defined cutoff (e.g., m/z values 998.472 and 998.502). However, isotope m/z values, e.g., m/z values 999.481, an isotope of m/z value 998.472, were frequently dismissed as unimportant compared to the unsupervised CAERF approach (Fig. 4.16B, row 1). Additionally, many more noisy features received a high score (Fig. 4.17A, compare Fig. 4.14) in the RF only approach. Also in the semi-supervised CAERF approach some isotope m/z values were not retained.

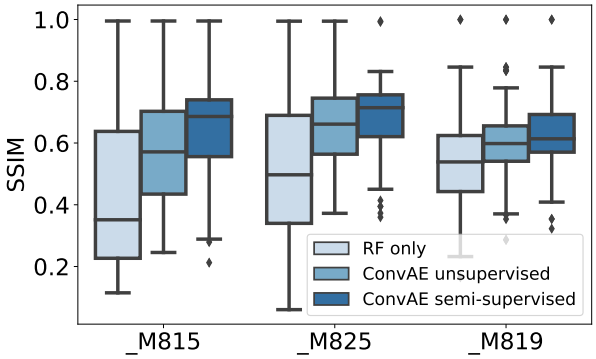


Figure 4.18: Qualitative analysis of exemplary runs of combined convolutional autoencoder and random forest (CAERF) approaches and random forest (RF) only approach. Boxplots illustrate the structural similarity index measure (SSIM) of all identified hypoxia-associated features (156 in RF only versus 180 in unsupervised CAERF approach versus 120 in semi-supervised CAERF approach) to the highest-ranked m/z value 998.472 per sample. A higher score indicates a higher degree of similarity to the reference feature. This figure was first published in Bitto et al. [111] with naming of the samples being adapted for this thesis for consistency.

For a systematic comparison, a similarity score for every m/z value associated with hypoxia and the highest ranked RF m/z value 998.472 was calculated using the structural similarity index measure (SSIM). If all found features are positively correlated to the hypoxia annotations, one would expect a high overall SSIM. Fig. 4.18 shows that the found 180 features of the CAERF approach were more similar than the 156 features of the RF only approach, considering all three samples. Given that tree-based feature importance metrics attempt to extract discriminative features, these methods may prioritize features that are unrelated to the top-ranked feature(s). The highest SSIM score was reached by the semi-supervised CAERF approach, producing the most alike set of features.

4.2.2.2 Quantitative Results of Ten Runs Each

Importance of Latent Space Size

As described in Section 4.1, the optimal latent space size was approached using R^2 adjusted using 10 experiments each for every latent space configuration. To estimate the latent space size, the following data characteristics were considered:

- Up to 4 raw m/z values represent isotopes. Isotopes are neighboring m/z values that are off by one, which share the same chemical characteristics. They will depict similar spatial patterns and thus should be easily summarizable (see Fig. 4.19).
- Also m/z values that reflect potential mass shifts are likely to consolidate into the same latent space feature(s).
- Using trypsin for protein digestion will lead to additional m/z values, which are expected to show high correlations.

Considering this information, latent feature space z^2 was initially set to 256 and steadily decreased to 8. Fig. 4.20A shows that according to R^2 adjusted, the best performance was reached at a latent space size of approximately 64. In particular, a low latent space size of 8 was not reliably retaining a sufficient amount of hypoxia features. A higher latent space size, e.g., of 128, apparently introduced too many redundancies in terms of correlating latent space features, and was not able to improve the metric score further. Of note, also the larger reconstruction error of models with a latent space size of 8 demonstrated a poorer performance compared to the other configurations. The approximate reconstruction

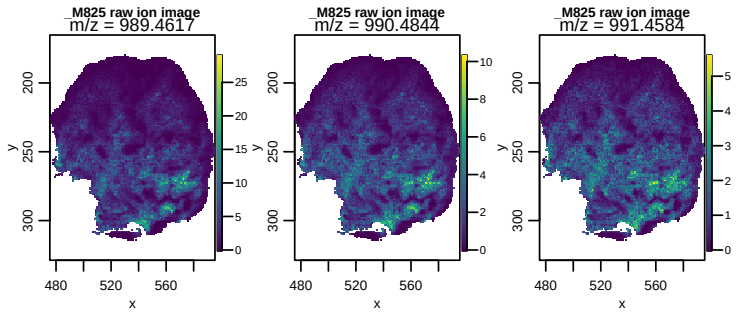


Figure 4.19: Isotope mass-to-charge ratio (m/z) values (989.4617, 990.4844, 991.4584) of the same peptide, shown are (non-normalized) ion images of one sample. This figure was first published in Bitto et al. [111] with naming of the samples being adapted for this thesis for consistency.

error for all remaining configurations (32, 64, 128, 256) only differed at the time point at which the models reached convergence.

As the described characteristics are the same for the unsupervised CAERF and the semi-supervised CAERF approach, a latent feature space configuration of $z^2 = 64$ was employed.

Reproducibility of Results

All three approaches, i.e., the RF only, the unsupervised CAERF and the semi-supervised CAERF approach, were run ten times to evaluate whether the overall results led to similar conclusions presented earlier in this section. Fig. 4.20B visualizes that similar SSIM scores to those in one individual run were achieved. Fig. 4.20C also outlines that the overall number of features associated with hypoxia was lowest in the semi-supservised CAERF approach. This can be linked to higher specificity of the latent feature associated by hypoxia, as a result of incorporating hypoxia labels during the training of the AE. One important observation is, that in some cases the CAERF runs achieved relatively low SSIM scores, although achieving comparable results for R^2 adjusted. Inspecting those runs showed, that the highest ranked latent feature for hypoxia was sometimes negatively correlated to the hypoxia annotations, although also positively correlated latent space features existed. This limitation can be attributed to the general behavior of tree-based feature importance metrics. Although this event was rare, one potential solution could involve not relying solely on the ranking of feature importance scores.

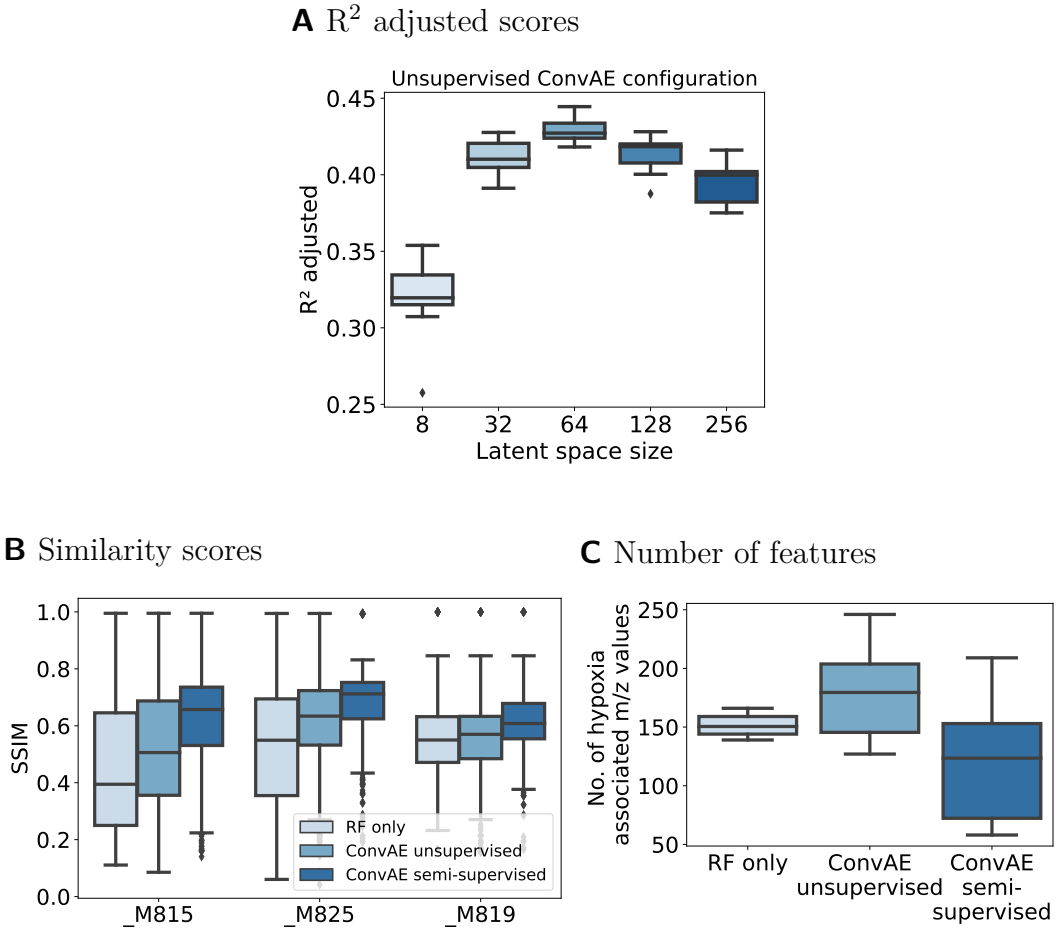


Figure 4.20: Quantitative analysis of 10 runs each. (A) Latent space configurations were compared using R^2 adjusted of the fitted regression models using ten unsupervised convolutional autoencoder and random forest (CAERF) runs per configuration. (B) Boxplots illustrate the structural similarity index measure (SSIM) of all identified hypoxia-associated features to the highest-ranked mass-to-charge ratio (m/z) value 998.472 per sample in ten individual runs each. A higher score indicates a higher degree of similarity to the reference feature. (C) Boxplots show the number of hypoxia-associated m/z values that was identified by all three approaches in the ten runs. This figure was first published in Bitto et al. [111] with naming of the samples being adapted for this thesis for consistency.

Possible Adjustments to Random Forest Only Approach

RFs were not specifically designed to identify all relevant features for a given classification or regression task. Therefore, it was assessed how an extension of RF, the Boruta algorithm,

performs on the task of identifying relevant features of hypoxia from spatial omics data. Rather then inputting the 18,735 feature into the RF only approach, Boruta was applied as initial feature selection method. Considering 10 independent runs, approximately 989 m/z values [95% confidence intervall (CI), 916.5–1,062.0] were marked as relevant, and 1,226 more as tentatively relevant [95% CI, 1,083.5-1,368.9]. Each run performed 10 iterations, with a higher setup increasing the number of relevant features further (e.g., 1,644 confirmed and 774 tentatively relevant features in a run with 100 iterations). This preselection of features is too coarse to provide meaningful insights into hypoxia and would produce too many false positive peptide candidates when linked to LC-MS/MS masses. The reduced Boruta feature set yielded to similar results when provided to the RF only model as described earlier in this Section.

In summary, the proposed CAERF approach showed more powerful aggregation capabilities by reducing the high-dimensional feature set to a relevant set of hypoxia candidates than the RF only or the Boruta algorithm.

4.3 Spatial Transcriptomics

Signals of tumor hypoxia might also be apparent in gene expression of spatial transcriptomics platforms. Compared to MSI, the available platforms can be considered more accessible, as they typically provide tools for directly outputting gene expression data per spatial spot. For example, Space Ranger is a software developed by 10x Genomics as support for their Visium Spatial Gene Expression platform [118]. Given some FASTQ files and a microscope slide image (like H&E stains), the software will take care of the alignment of the sequencing data to a reference genome, registering the microscopy image to the the spatial data using fiducial markers and counting the molecular barcodes associated with its transcripts to determine gene expression. Still, the underlying difficulties of high-dimensional, correlated data remain. While in MSI, m/z values might be highly correlated because they belong to the same peptide, in SPT, high correlation might arise from co-expressed genes. Barcode technologies come in particular with one main additional challenge for data analysis, i.e., the alignment of spots often does not follow a strict grid arrangement. For Visium, spots are arranged in an offset pattern referred to as "orange crate packing", becoming apparent when coordinates are sorted according to x and y coordinates. This is particular challenging for convolutional layers with a small kernel size, as up to about half of the pixels will not contain any information.

Additionally, the aggregation of multiple samples makes it necessary to derive a common set of features, similarly to MSI. However, for SPT data this only requires a common intersection set of all genes.

Both issues are addressed in Section 4.3.1. Results of the unsupervised CAERF approach, RF only approach and semi-supervised CAERF approach are presented in Section 4.2.2.

4.3.1 Methods

Sample preparation and SPT experiments were not performed by myself, with protocols briefly being sketched in Section 3.1.1. This section describes the pre-processing of SPT data and adjustments to the CAERF approach.

4.3.1.1 Pre-Processing of SPT data

In the following, italic names in brackets denote the corresponding parameter in the described software or libraries.

Samples were first independently pre-processed using Space Ranger, resulting in a data set that contains gene expression data and the co-registered H&E stain. Next, the consecutive FI was co-registered. The process of co-registration as outlined in Section 4.1.1, which primarily relies on a common tissue shape, encountered obstacles from the alternating pattern of SPT data. Therefore, I decided to not directly align the FI to the data but instead register the FI to the H&E image. SpaceRanger provides two downsampled versions of the H&E, of which the *tissue_lowres_image* with 600 pixels in its largest dimension was chosen as *fixed* image. Apart from defining a different image modality as fixed image, the overall procedure of co-registration remained unchanged. The retrieved hypoxia annotations from the co-registered FI were then linked to the SPT data in a two-step approach: First, the center position of every spatial spot in the downsampled H&E image was estimated. This can be achieved by multiplying the center positions of every spatial spot in the full resolution H&E image (see Table. 4.2), with the known scaling factor of the downsampled H&E image (*tissue_lowres_scalef*).

Table 4.2: Example exert of tissue position file (output file of Space Ranger). Rows show the spatial barcodes. Values *array_row* and *array_col* denote the coordinates of a spot (rows: from 0 to 77, columns: even numbers from 0 to 126 for even rows, and odd numbers from 1 to 127 for odd rows), values $pxl\_row\_in\_fullres$ and $pxl\_col\_in\_fullres$ represent the pixel coordinates of the center of the spot in the full resolution image [137]. Shown are exemplary values from one SPT sample.

Barcode	in_tissue	array_row	array_col	pxl_row_in_fullres	pxl_col_in_fullres
ACGCCTGACACGCGCT-1	0	0	0	26183	2984
TACCGATCCAACACTT-1	0	1	1	25999	3302
ATTAAAGCGGACGAGC-1	0	0	2	25816	2983
GATAAGGGACGATTAG-1	0	1	3	25633	3302
GTGCAAATCACCAATA-1	0	0	4	25450	2983
TGTTGGCTGGCGGAAG-1	0	1	5	25266	3302
GCATCCTCTCCTATTA-1	0	0	6	25083	2983
GCGAGGGACTGCTAGA-1	0	1	7	24900	3301

Second, the pixel coordinates of the co-registered FI were mapped to the derived spot coordinates in the downsampled H&E using nearest neighbor interpolation. Pixels

containing only background (indicated by a pixel value of 0.0) were not mapped. However, as the data structure of SPT is sparse, many coordinates of a grid are missing, e.g., *arrow_row*, *arrow_col* = (0,1). Instead of omitting hypoxia pixel coordinates for which no matching H&E pixel coordinate could be found, the sparse coordinate data structure was densified to a grid-like structure. First, the data structure was extended to encompass all missing coordinates for *arrow_row* and *arrow_col*. The corresponding pixel coordinates for the downsampled H&E image were then interpolated via a radial basis function (see Python Code Snippet 4.2). The gene expression data itself was NaN-filled. This allows to

Code Snippet 4.2: Extension of SPT pixel coordinates

```
import pandas as pd
from scipy.interpolate import RBFInterpolator

def interpolate_lowres_x_y(spt_data):
    x_range = range(spt_data.obs['x'].min(), spt_data.obs['x'].max())
    y_range = range(spt_data.obs['y'].min(), spt_data.obs['y'].max())
    all_combinations = pd.DataFrame([(i, j) for i in x_range for j in y_range],
                                    columns=['x', 'y'])

    joined = all_combinations.merge(spt_data.obs, left_on=["x", "y"],
                                    right_on=["x", "y"], how="left")
    interpolate_fu = RBFInterpolator(spt_data.obs[["x", "y"]],
                                     spt_data.obs[["x_lowres", "y_lowres"]])
    interpolated = pd.DataFrame(interpolate_fu(joined[["x", "y"]]),
                                columns=["xi_lowres", "yi_lowres"])
    all_coords = pd.concat([joined, interpolated], axis=1)
    return all_coords
```

retain all hypoxia pixel annotations, which would otherwise be significantly reduced in size (see Fig.4.21)

Then, the SPT data of every sample was normalized using the *sctransform* procedure via the Seurat package. Filtering was applied by means of the Scanpy library. Precisely, only spots with a minimum count of 500 and a maximum count of 35000 were retained [*scanpy.pp.filter_cells(min_counts=500, max_counts=3500)*]. Additionally, only genes were kept that were present in at least 20 individual spots [*scanpy.pp.filter_genes(min_cells=20)*]. Following, all samples were combined by only retaining the intersection set of common genes. The pre-processing steps are shown in Supplementary Material, Code Snippet 7.12.

Of note, for normalization, the in-built total-count normalization method of Scanpy

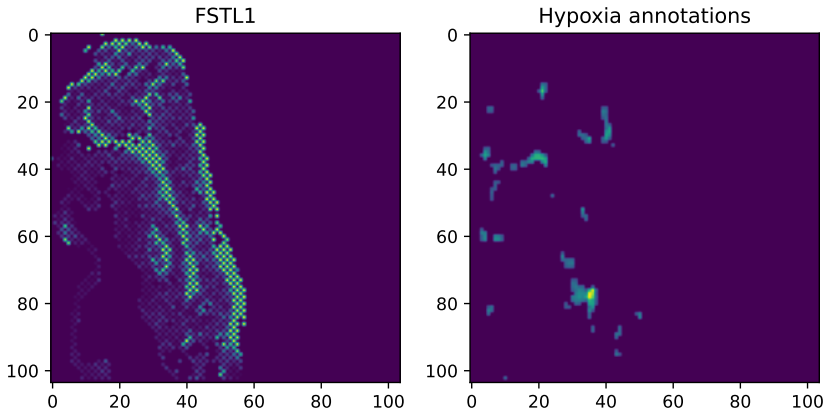


Figure 4.21: Example of gene expression image (FSTL1, left) and the derived hypoxia annotations (right). The gene expression image depicts visible spots due to its orange crate packing. For co-registration with the hypoxia image, the sparse data structure was densified to allow to retain all pixels from hypoxia annotations.

(*normalize_total*) and the *sctransform* procedure of Seurat were considered. Although both methods were primarily designed for single-cell data, they are commonly applied to spatially-resolved gene expression data like for the Visium kit used in this thesis. In the total-count approach, each spot is divided by the total counts, aiming to account for differences in sequencing depth. *Sctransform* is a more elaborate procedure in which a generalized linear model, more specifically a negative binomial regression model, is constructed for every gene individually [119]. It further performs variance stabilization across genes. Overall, the choice of normalization altered some of the genes associated with hypoxia as well as results of correlation analysis. To date, no gold standard for SPT pre-processing has been established. However, with the expectation of non-linear relationships between genes, the *sctransform* procedure seems to be the preferable choice for pre-processing. The choice is supported by findings from Choudhardy and Satija, which found that other commonly applied generalized linear models like Poisson error models show evidence of overdispersion for genes [138].

4.3.1.2 Autoencoder Setup

First experiments with the CAE architecture described in Section 4.1.2.1 revealed that the CAE was not able to learn meaningful latent representations. This could be ascribed to the orange crate packing of the data. Given a patch size of 3 ($x = 3$), at least 4 pixels

out of 9 contain no information about the data, referred to in the following as zero pixels. Therefore, a weighting of data pixels on the reconstruction error was introduced. The weighting of pixels was directly integrated into the loss function shown in method *do_step()* of the class *WeightedAETrainer* in the Supplementary Material, Code Snippet 7.7.

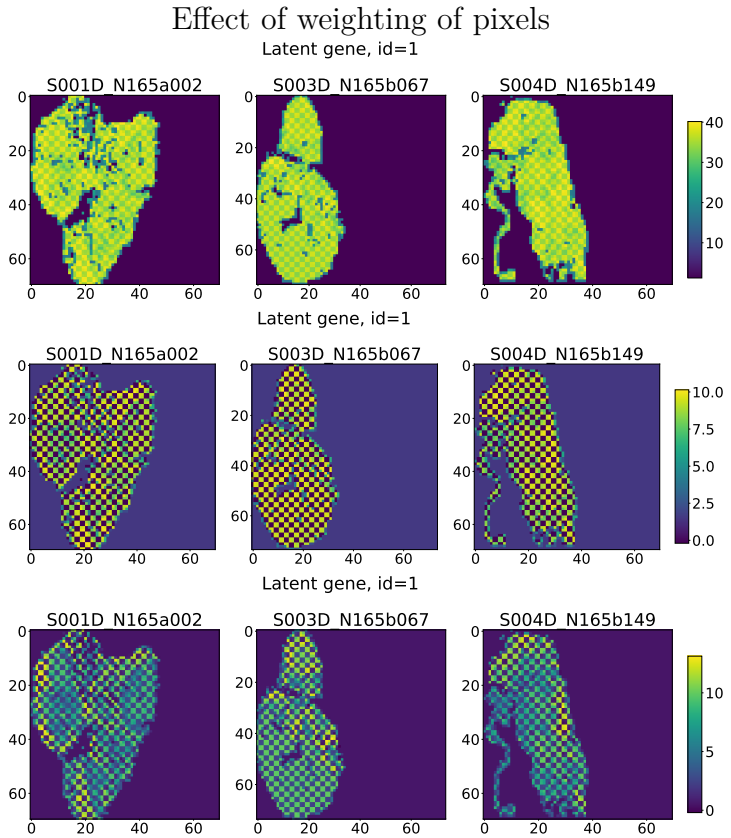


Figure 4.22: Effect of orange crate packing. Shown are patches encoded with the proposed convolutional autoencoder (CAE) architecture, utilizing kernels with $size = 2 \times 2$ and $strides = 1$ (b = 1) in the second hidden layer with patch size $x = 3$. Top: No weighting of pixels. Middle: Data pixels are weighted 10 times higher than zero pixels. Bottom: Data pixels are weighted 100 times higher than zero pixels.

Fig. 4.22 shows how the weighting affects the latent representation when the data pixels were weighted not at all (top), 10 times higher than the zero pixels (middle), 100 times higher than the zero pixels (bottom). With weighting, some structure within the tissue becomes apparent. However, the figure also highlights that the latent space still preserved zero pixels. In order to enforce that the autoencoder learns more global structures, the stride size of the second hidden layer was increased from 1 to 2 (b = 2). As a consequence,

the patch size was adjusted from 3 to 4, allowing the kernels to traverse a patch entirely. Interestingly, the adaption of the kernel size lead to a high variability in overall intensity levels between samples. This was mitigated, when the kernel setup was flipped in the two hidden layers. The final configuration (see Section 4.1.2.1) was set to: x = 4, y = 10913, a = 2, z^1 = 1024, b = 0, z^2 = 64–256. For the unsupervised CAERF, a 100 fold weighting of data spots to zero pixels was utilized. For the semi-supervised approach, the weighting was reduced to 10 fold to balance the overall loss of the reconstruction and semi-supervised error.

4.3.1.3 Sample Size

SPT data of untreated tumor model SAT and CAL33 were pre-processed and split into patches of size 4×4 pixels. Likewise, the corresponding hypoxia annotations were split into patches of size 4×4 pixels. All data was scaled to a range between [0, 1]. Overlapping patches with a step size of 1 were generated. Patches containing no SPT data but only background were removed. The CAE (unsupervised and semi-supervised) was then trained on 13,567 patches from three SAT samples. For validation 393 patches from one CAL33 sample were utilized.

The three samples used for training the AE, were also utilized for evaluating the performance on the subsequent RF regression models. The RF regression model was built using overlapping patches of stride 1. Like for MSI, the number of non-hypoxic patches was downsampled. Given the low number of hypoxic spots, all patches with at least 0.05 mean hypoxia expression were chosen in addition to 50% non-hypoxic patches, resulting in a total of 2,482 patches. Performance and feature importance metrics were evaluated using 10-fold cross validation with a test size ratio of 33%.

4.3.1.4 Random Forest Setup

The same setup for the RF regression models were utilized as described in the MSI methods part.

4.3.2 Results

From the four SPT samples, a total of 10,913 common genes were derived. For every SPT sample, hypoxia annotations from consecutive FIs were extracted (see Fig. 4.23). In the following, the three approaches were evaluated by qualitative results of individual runs and quantitative results of multiple runs.

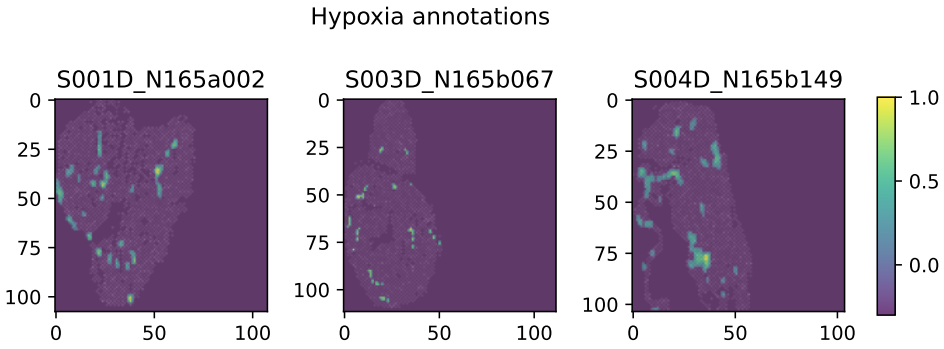


Figure 4.23: Hypoxia annotations of individual spatial transcriptomics (SPT) samples. Yellow = high degree of hypoxia, dark blue = no hypoxia.

4.3.2.1 Qualitative Results of One Exemplary Run Each

Results of Unsupervised Convolutional Autoencoder Run

The CAERF was trained with a latent space size z^2 of 64. Out of the 64 latent features, #37 exhibited the highest RF feature importance for hypoxia (see. Fig. 4.24). According to the proposed recovery method, 162 genes were associated with this feature, using a Spearman correlation coefficient cutoff value greater than 0.975. Some exemplary genes are shown in Fig. 4.25A and 4.25B. All genes are listed in the Supplementary Material in Section 7.4.1.

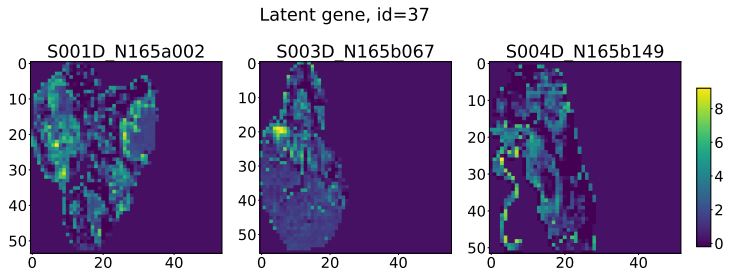
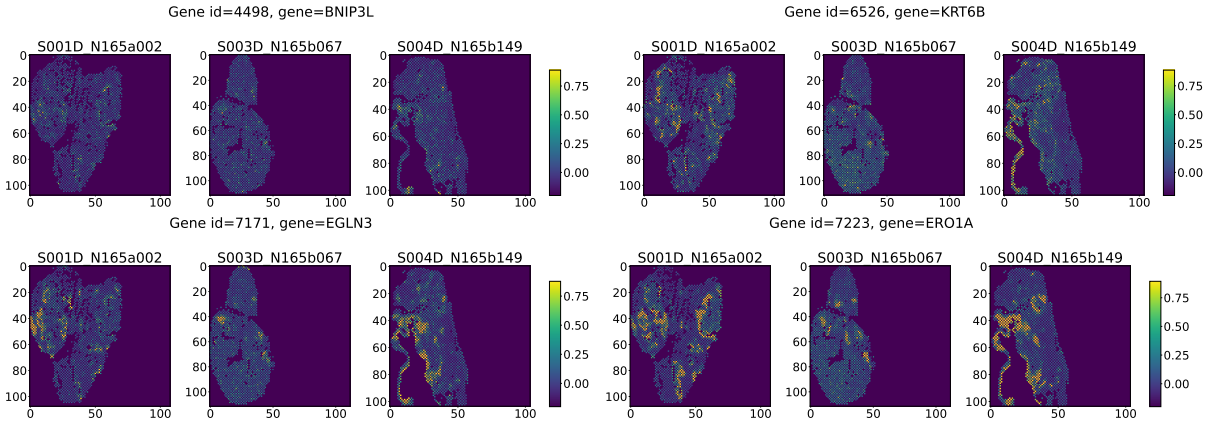


Figure 4.24: Visual representations of encoded spatial transcriptomics (SPT) samples in latent space for latent space feature #37 associated with hypoxia according to random forest (RF) feature importance.

A Common genes found by both CAERF approaches and RF only



B Genes found by unsupervised CAERF approach

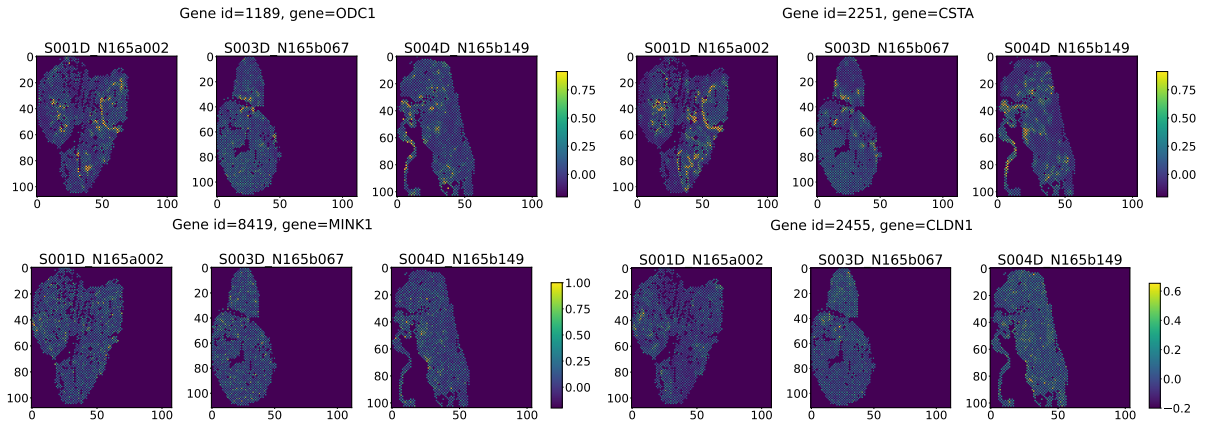


Figure 4.25: Exemplary genes associated with hypoxia that (A) were found by the both convolutional autoencoder and random forest (CAERF) approaches and the random forest (RF) only approach or (B) were distinctively found by the unsupervised CAERF approach.

Considering the high amount of genes that was recovered with a strict cutoff value of 0.975, the specificity of the derived latent feature and recovery method were investigated. Therefore, an arbitrary feature with a modest correlation to the hypoxia annotations from the latent features was chosen. Latent feature #32 was positioned between the 17th and 20th place in the feature importance ranking. The recovery method assigned 89 features to the latent feature, of which only seven were in common with the 162 features of the highest ranked latent feature #37. Although few in number, these seven features (two of them shown in Fig. 4.26) rather depicted noisy associations.

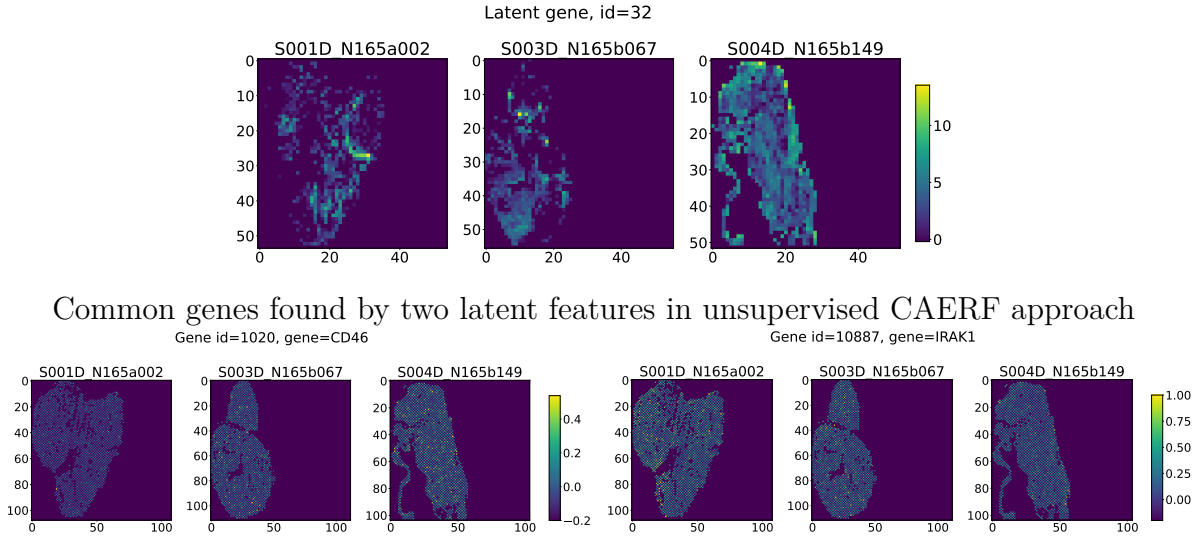
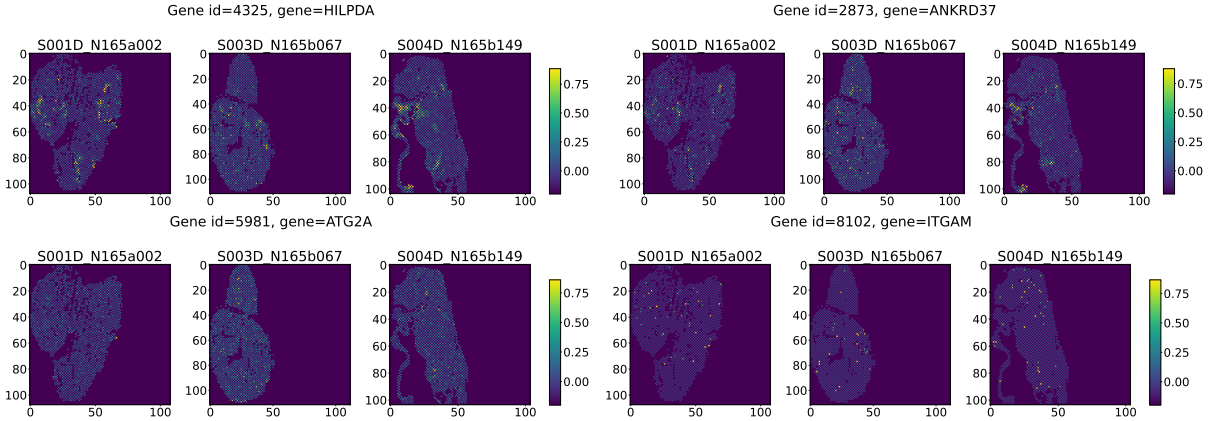


Figure 4.26: Latent feature #37 (top) and exemplary genes (bottom) that were found by the recovery method in the unsupervised convolutional autoencoder and random forest (CAERF) approach to contribute to both, feature #37 (high hypoxia association) and feature #32 (low hypoxia association). The genes likely denote noisy associations to hypoxia as they depict no clear pattern.

Results of Random Forest Only Run

In the RF only approach, all 10,913 genes were used as input for the regression task. In an exemplary RF only run, 192 genes were found to be associated with hypoxia. The cutoff was set so that a comparable number of features as in the CAERF approach was picked. Therefore, all features were chosen that reached at least a fourth of the highest score in all cross validation runs. In 4 out of 10 cross validation runs, gene *HILPDA* achieved the highest feature importance score. Some exemplary genes are shown in Fig. 4.25A and 4.27A. All genes are listed in the Supplementary Material in Section 7.4.2.

A Genes found by RF only



B Genes found by semi-supervised CAERF approach

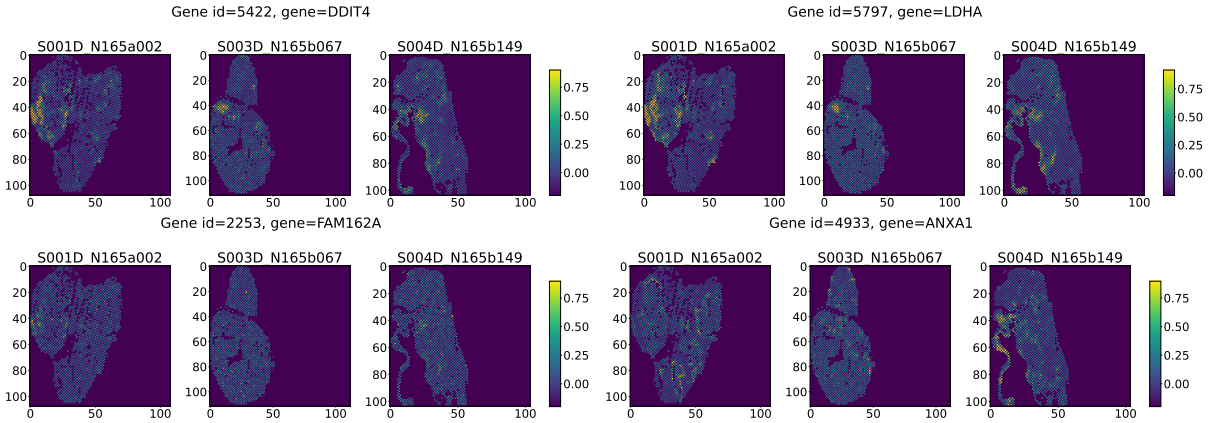


Figure 4.27: Exemplary genes associated with hypoxia that (A) were distinctively found by the random forest (RF) only approach or that (B) were distinctively found by the semi-supervised CAERF approach.

Results of Semi-Supervised Convolutional Autoencoder Run

In the semi-supervised CAERF run, the number of features which was found to be associated with hypoxia increased to a total of 234. Some exemplary genes are shown in Fig. 4.25A and 4.27B. All genes are listed in the Supplementary Material in Section 7.4.3.

Comparison of CAERF and RF Only Approaches: Hypoxia Associated Genes

In total, only 12 genes were common in the unsupervised / semi-supervised CAERF and the RF only approaches (four of them are shown in Fig. 4.25A). Interestingly, the RF only approach ranked many features as important which were apparently just noise. For

example, genes *ATG2A* and *ITGAM* depicted high expressions only in individual pixels (see Fig. 4.27A) but were ranked highest in several cross-validation runs. By contrast, the unsupervised CAERF approach picked up genes depicting some small intensity spots but otherwise showing low variability (e.g., MINK1, in sample S004D_N165b149 see Fig. 4.25B). Although some of those genes may be considered as noisy associations, far fewer were affected compared to the RF only approach. While in the semi-supervised CAERF the overall number of hypoxia-associated genes increased further, many of the noisy associations of the unsupervised CAERF and the RF only approach were dismissed. Also, some gene candidates that depicted some overlap with hypoxia annotations and that showed up in the RF only but not in the unsupervised CAERF approach, were recovered in the semi-supervised CAERF run, e.g., *ADM* and *FAM83A*. Several of the gene candidates derived from the CAERF approaches were associated with hypoxia previously. For instance, *ADM, BNIP3L, EGLN3, SLC2A1, ERO1A* (or its synonym *ERO1L*) and *NDRG1* were examined previously by Toustrup et al. for their hypoxia gene signature [46].

For a comparison of all features, again a SSIM analysis was conducted. As reference, gene *BNIP3L* was used, which was associated with hypoxia by all three approaches. Fig. 4.28 illustrates a wide range of feature similarity scores in the RF only approach, likely due

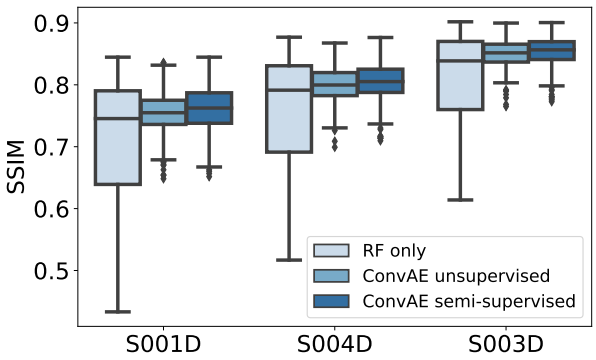


Figure 4.28: Qualitative analysis of exemplary runs of convolutional autoencoder and random forest (CAERF) approaches and random forest (RF) only approach. Boxplots show the structural similarity index measure (SSIM) of all identified hypoxia-associated features in RF only (192 features) versus unsupervised CAERF approach (162 features) versus semi-supervised CAERF approach (234 features) using *BNIP3L* as reference per sample. The gene *BNIP3L* itself was removed for analysis.

to many noisy associations. In comparison, the boxplots of the unsupervised CAERF run are substantially narrower. However, the upper quartiles also indicate that the RF only approach identified some promising gene-candidates that the unsupervised CAERF approach missed. The upper quartiles of the semi-supervised CAERF approach suggest that some further genes with a high SSIM score were detected without any additional noisy associations.

4.3.2.2 Quantitative Results of Ten Runs Each

Importance of Latent Space Size

The optimal latent space size was again approached using R^2 adjusted using 10 experiments each for every latent space configuration (Fig. 4.29A).

R^2 was found highest with a latent space configuration of 64. A latent space size of 8 missed relevant features of hypoxia. A higher latent space size (128, 256) may deliver results comparable to those obtained with 64 features, but will likely produce higher correlating latent space features. As a result of the findings, a latent feature space configuration of $z^2 = 64$ was utilized for the unsupervised CAERF and the semi-supervised CAERF approach.

Reproducibility of Results

Each approach was run ten times to evaluate whether the findings from qualitative runs were reproducible. Fig. 4.29B confirms that the best overall SSIM score was achieved by the semi-supervised CAERF approach, followed by the unsupervised CAERF approach. The cutoff values in all approaches were selected to achieve a comparable number of features. However, it turned out that the number of genes in the semi-supervised approach was significantly greater than that in the other approaches (see Fig. 4.29C). Therefore, it is likely that a stricter cutoff value would lead to even higher SSIM scores in the semi-supervised CAERF approach.

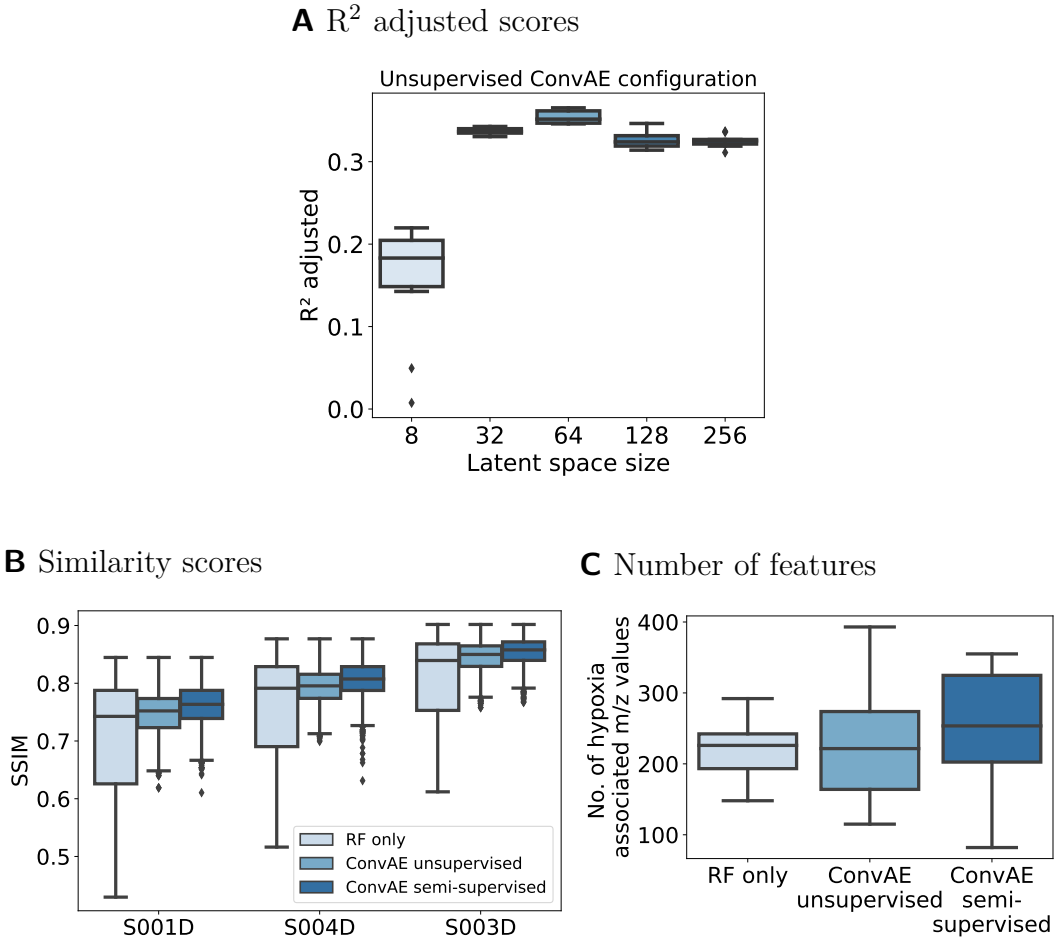


Figure 4.29: Quantitative analysis of 10 runs each. (A) Latent space configurations were compared using R^2 adjusted of the fitted regression models using ten unsupervised convolutional autoencoder and random forest (CAERF) runs per configuration. (B) Boxplots illustrate the structural similarity index measure (SSIM) of all identified hypoxia-associated features to the highest-ranked gene *BNIP3L* per sample in ten individual runs each. (C) Boxplots show the number of hypoxia-associated genes that was identified by all three approaches in the ten runs.

4.4 Discussion

For both spatial omics modalities, MALDI MSI and sequencing-based SPT, it was demonstrated, that the technologies are capable of measuring signals that overlap with hypoxia annotations in heterogeneous tissue of HNSCC xenograft models. The workflow of the CAERF approaches was designed to reduce the high-dimensional feature space of spatial omics while retaining hypoxia-associated features and explainability. It was outlined that CAEs can provide a valuable companion to feature selection methods when processing highly correlated spatial omics data. With the help of the proposed recovery method, original features that contributed to a hypoxia-associated latent feature could be identified. The resulting feature set was more coherent in terms of the features' similarity as compared to features from RF only models. More promising, in case of MSI, the feature set could be linked to masses from LC-MS/MS experiments to derive actual peptide candidates associated with hypoxia. One specific aspect of the MSI data used here was that it consists of tryptic peptide information, increasing the multicollinearity of molecular features even further. Among the found gene and peptide candidates, many were linked to hypoxia before, reinforcing the reliability of the proposed CAERF approach. In the following, alternative design decisions, limitations and possible improvements are discussed.

Methodologically, I initially considered also other machine learning approaches. For example, instead of a combinational CAE and RF approach, also a CNN might be used for predicting hypoxia from spatial omics data. Nevertheless, the task of identifying which molecular features contributed to hypoxia would have remained the same for any other model. As will be outlined in Section 5.2, autoencoders may be particular intriguing when the latent space can be utilized for subsequent analysis. The RF regression model might as well be exchanged by another feature selection method. I decided to use RF models as they enable the modelling of complex relationship between features and thus allow for a decent comparison of dimensionality-reduced and unreduced spatial omics data. Additionally, a variety of intuitive feature importance metrics are readily available for RF. The most commonly applied RF feature importance measures, i.e., impurity-based importance (IBI) and permutation importance (PI), were examined. Both, IBI and PI, were found to be biased when processing highly correlated predictor variables. For IBI, Nicodemous et al. found that it gives higher weight to uncorrelated although

non-predictive features rather than correlated ones [73]. On the contrary, PI was associated with overestimating the importance of correlated predictors [74]. Importantly, all of these findings were derived from classification tasks, but they likely extend to regression tasks. This assumption is grounded in the fact that the underlying algorithms remain unchanged with the exception of the scoring metric, e.g., using MSE instead of Gini index as impurity measure. The existence of fully uncorrelated features is improbable in a spatial omics setup; instead, pairs of features are expected to correlate with one another, differing only in the strength of correlations. Therefore, I decided to apply IBI, considering also its higher computational efficiency on large datasets. Overall, the results demonstrated that IBI identified some relevant features for hypoxia in the uncompressed data, but among the top-ranked features, many more noisy than true associations were found. In comparison, the highest-ranked latent feature of the compressed data delivered more true positive correlations to the hypoxia annotations. Also the Boruta algorithm was examined as it was especially designed for identifying *all-relevant* predictors in a feature selection task [76]. Nevertheless, for MSI, the number of relevant features remained way too high to allow for subsequent analysis. One other frequently used method to explain the results of machine learning methods includes Shapley values. In theory, Shapley values do not assume feature independence and therefore might be applicable for a highly correlated features space. However, common implementations do, e.g., Kernel SHAP [101, 139], to cope with the computational complexity that would otherwise grow exponentially [140]. Therefore, this alternative approach was disregarded.

Autoencoders were used for dimensionality reduction of spatial omics data before. Among the first, Thomas et al. used AEs to reduce the MSI data of mouse brain to 15 latent features, using individual pixels as input. They visualized the latent features and showed that some of them depicted brain regions, while others were considered noise [86]. Abdelmoula et al. showed the effect of variational AEs on various 2D and 3D MSI datasets. Like Thomas et al., they implemented a pixelwise approach with a restrictive latent space of size 5, resulting in latent features approximating tissue anatomy. They also proposed an algorithm to derive so-called informative m/z peaks, i.e., original features that were similar to the pattern of the latent feature of interest, based on a threshold analysis on the weight parameters of the AE. Matsuda et al. used a sparse AE architecture to compress 468 features of human skin structures to 20 latent features from TOF secondary ion mass spectrometry imaging (TOF-SIMS) [87]. Gardner et al. showed the usage of CAE on

TOF-SIMS data from tumor spheroid samples [141]. While they used patches instead of pixels like in this work, the actual architecture (e.g., layers, patch size, activation function among others) differed. Again, the latent feature size of 25 was capturing primarily histological features. Further related work includes Li et al. which reduced MS (not MSI) data to 256 latent features by means of a denoising AE [142]. They also combined the AE approach with RF models (among others), albeit for the classification of Listeria. Classification tasks on MSI data were performed on various deep learning architectures [143], such as CNN for tumor classification [144] or recurrent neural networks for detecting cancerous regions [145].

For SPT, AE-based approaches for dimensionality reduction have been used less commonly to date. On reason for this might be that SPT platforms with transcriptome-wide capabilities have only recently became available. Probably, there may be less urgent need, given that readily available pre-processing pipelines (like Space Ranger) and subsequent analysis software (like Seurat or Scanpy) allow for basic exploration of the data. The most commonly conducted analyses for SPT data involve spatial differential gene expression analysis, deconvolution and resolution enhancement through the integration of scRNA data [146, 147]. Among the deep learning approaches suitable for dimensionality reduction purposes, Xu et al. suggested a two components model, consisting of a deep mask autoencoder and a variational graph autoencoder to learn a low-dimensional latent representation from SPT data [148]. Although the precise number of created low-dimensional embeddings was not mentioned, it was proposed for clustering, visualization, trajectory inference and batch integration. Similarly, Dong et al. developed a graph attention autoencoder to learn low-dimensional embeddings from SPT data [149]. The authors showcased that the model's embeddings were able to segment tissue regions in a coronal mouse brain.

In this thesis, the main focus was to retain hypoxia-related features, which exhibit less dominant signals than tissue morphology. Therefore, it was shown how to approach an optimal latent space size. In all mentioned AE-based works, a precise description on how the latent space setup was established is missing. Only Abdelmoula et al. stated that their latent space configuration was found empirically. They recommended to consider the reconstruction error for finding the optimal latent space. However, as discussed in the MSI part, different latent space configurations may yield comparable reconstruction errors during unsupervised training but the latent space might not be equally well suited

for subsequent analysis. In this work, R^2 adjusted of the trained RF models was factored in to evaluate which latent space maintained the most valuable hypoxia information when the number of features was penalized. Also in contrasts to other publications, this work highlighted how data from multiple samples can be processed effectively. Using multiple samples prevents the AE to overfit on the characteristics of individual samples and thus will produce more robust latent features. However, finding a common set of features may imposes some additional challenges as outlined in the MSI section. These challenges were overcome by sticking close to the raw m/z values and relying on the aggregation capabilities of the AE.

Generally, MSI data can be considered less accessible than SPT data, given that it comprises m/z values. Therefore, MSI is often complemented by some kind of tandem MS data to derive peptide information, like in this work. Kassuhn et al. mapped MSI m/z values to complementary nano-LC-MS/MS data from adjacent tissue sections by considering peptides with the lowest mass difference to be a match [150]. Also Hoffmann et al. correlated MSI m/z values to LC-MS/MS data from HNSCC patients, but limited the analysis to the 10 most characteristic m/z values found in tumor tissue [151]. Studies on the investigation of tumor hypoxia by means of MSI are rare. Djidja et al. analyzed hypoxia in MSI data (among other data) of 4T1 models, i.e., a mouse model for studying breast cancer [152]. They were able to identify 18 proteins by combining MSI data with LC-MS/MS experiments. Among them, five proteins were significantly associated with hypoxic regions from immunohistochemistry stainings as determined by a Student's t-test. In that work, a Student's t-test was statistically applicable given the low number of protein candidates. They also identified proteins from microdissected hypoxic regions, of which several were in line with the ones found here (e.g., LDHA, PGK1, LMNA, PKM, ANXA1, ALDOA). An indirect approach was pursued by Mascini et al. which demonstrated that tissue of pimonidazole-injected animals allows to trace signals of the hypoxia marker [153]. Of note, also the xenograft models in this work were injected with pimonidazole before excision of the tumor, but with a much lower dose (0.4 mg/g versus 0.1 mg/g [59]). Despite the fact that pimonidazole signals might obscure the true molecular signals of hypoxia, my observations showed that our data did not contain dedicated signals of pimonidazole.

The mapping of MSI m/z values to LC-MS/MS peptides should be considered as

initial peptide candidate list, which requires further evaluation, e.g., via immunohistochemistry. As shown in Table 4.1, one mass might be mapped to several different peptides. This can be ascribed to the lower mass resolving power and mass accuracy of TOF-based MS in comparison to other mass analyzers [32]. With next-generation technologies like FTICR, this issue will likely to be overcome [34]. However, the proposed workflow diminishes the likelihood of random matches by only considering peptides for which at least two individual MS masses can be matched, and for which the corresponding ion images show some degree of correlation (Spearman correlation coefficient > 0.80). This parameter can be easily adjusted for more stringent results, depending on the number of false positives and false negatives that can be accepted.

While the CAERF approaches delivered promising unimodal peptide and gene candidates, several limitations need to be acknowledged. One general limitation is the usage of consecutive tissue slices to acquire the spatial omics data and the hypoxia annotations. Even when the two slices are direct neighbors, the underlying tumor morphology may differ. Additionally, some sheering, overlapping or disruption of tissue may occur at some point during sample preparation. Some of theses distortions may lead to misalignment of hypoxic cells, present in the fluorescence image but not in the corresponding tissue slice for which the spatial omics experiments are carried out or vice versa. Other distortions might affect proper co-registration of the two modalities. Since both, FIs and H&E images were required for this thesis, a choice had to be made between them. Although methods exist to combine multiple stains on one tissue slice, a combination of H&E and immunofluorescence is difficult due to their potential interaction with each other [154]. Therefore, the processing of consecutive tissue slices is a common challenge in omics data.

Even with a single tissue slice, co-registration of spatial omics data and fluorescence images would remain challenging, given the differences in resolution, contrasts and structures. In the registration process, a similarity transform was employed, enabling linear transformations like scaling, translation and rotation of the moving image. With regard to the results presented in Section 4.2 and 4.3, the co-registration appeared to perform adequately for its use case: Considering that the proposed workflow is utilizing overlapping patches and summarizing them later to a single mean value, precision errors during co-registration may not cause significant alterations to results. More elaborate

co-registration strategies for MSI and imaging data have been proposed. Patterson et al. suggested to capture multiple fluorescence images before and after MSI acquisition to then utilize laser ablation marks for co-registration [155]. Race et al. used representative images of the tissue morphology derived from the classification of a DenseNet model to co-register them to MSI [156]. However, their deep learning model was trained on H&E stains, where it was presumably easier to find distinguishable features that match with signals of MSI compared to fluorescence images. Cordes et al. published the open source application M^{2}aia [157] that expands the medical imaging toolkit [158] to support MSI data. Among various imaging pre-processing steps, M^{2}aia also supports image co-registration by integrating the elastix toolkit [106].

A related limitation is the absence of fully-annotated fluorescence images for hypoxia. Although a substantial amount of fluorescence images were stained, the raw images are of little use given the high variability in contrasts and artifacts. Using pixel classifiers to circumvent the need for fully annotated images, allowed to derive hypoxia annotations for some samples. However, also these pixel classifiers required re-training on individual tumor models and would benefit from more annotations.

Given that the AEs here were trained on samples of individual HNSCC tumor models (CAL33 for MSI, SAT for SPT), they might not generalize well to any HNSCC tumor model. This may be especially true for the semi-supervised CAERF approach, in which the AE may also memorize tumor model specific noise or outliers. However, the right sample setup could prove challenging as many different aspects may influence tumor hypoxia. For example, results on previously published work on the investigated tumor models here showed differences in hypoxia during treatment [59], which suggest some inherent distinctions between the phenotypes. Therefore, one must be careful not to undermine the relevant signals when adding additional samples for training the autoencoder. Of general note, it is important to consider that the training of NN-based models is subject to variability e.g., due to stochastic optimization algorithms or random weight initialization. As a result, the generated latent features may differ slightly for every run. It is therefore advisable to train multiple models and evaluate them with a proper metric, e.g., apply SSIM against a reference feature as proposed here. It was further shown, that spatial omics data may exhibit distinct characteristics (like the orange crate packing in SPT) that require adjustments to

the autoencoder architecture for effective training. This implies that for the application of new spatial omics data, fine-tuning of hyperparameters such as the patch size, the kernel size and the loss function (e.g., allow weighting of pixels) need to be considered.

Having highlighted the current limitations, several strategies will be explored for enhancing the CAERF approaches further. As part of future work I intend to evaluate the overall effect on results if different degrees of deformation are allowed for co-registration of FI and spatial omics data. However, finding the right deformation parameter setup may be tricky and may require manual interventions for specific samples. Overall, I expect that improvements will primarily influence the outcome of the RF regression models in all approaches alike.

Once more annotated FI are available, the sample size on which the CAE were trained will be steadily increased and CAERF findings concomitantly reevaluated. This also includes examining the effect of adding a more diverse set of HNSCC tumor models with known distinct treatment responses. Also the unsupervised and semi-supervised approaches will be re-investigated. The semi-supervised approaches proved effective in reducing noisy associations, indicated by the consistently higher SSIM score compared to the other approaches. Inevitably, this came at the cost of potentially dismissing true candidates (like isotopes in case of MSI). Nevertheless, compared to the results in MSI, the semi-supervised approach was not able to reduce the number of feature candidates in the SPT data, although the cutoff for the SPT data was already more restrictive than for MSI (Spearman correlation coefficient > 0.95 in MSI and Spearman correlation coefficient > 0.975 in SPT, respectively). Both CAERF approaches might therefore benefit by a subsequent filtering of candidates derived from the recovery method. For example, one may filter out genes which do not show a certain degree of variability in patches. Alternatively, a greater disentanglement of features in the latent space may be enforced during training. This could be achieved by an adapted loss function, for instance by penalizing a high correlation between latent features. Also the use of variational autoencoders, particularly the β extension, has been proposed to learn statistically independent latent features [159]. Yet, the degree of disentanglement could also shift to the opposite extreme. My early investigations on MSI data suggested that the adaption of the architecture to a variational autoencoder impacts the latent features considerably,

such that more abstract patterns rather than recognizable molecular patterns are preserved. Hence, an adaption of the loss function may be less invasive. Alternatively, the latent representations may be shaped by inputting a more balanced training set to the CAE. For example, an upsampling of patches that exhibit features of interest (like hypoxia) through data augmentation or a downsampling of non-hypoxic patches may be beneficial.

In summary, it was shown that CAEs are able to retain low intensity signals like tumor hypoxia. The results of the RF only approach indicated, that rankings of feature importance metrics should be considered cautiously if the research question requires identifying more than some relevant features. In these scenarios, AEs can be reliable companions of feature selection methods. The proposed recovery method of the CAERF approaches resulted in a promising set of peptide and gene candidates with potential link to tumor hypoxia. From these candidates, several showed up in both, MSI and SPT data, like *KRT16, LDHA, PGK1, ANXA1, KRT6B, TGM1.* The following chapter investigates strategies to combine both spatial omics modalities with the ultimate goal to derive multimodal gene-peptide biomarkers for hypoxia.

5 Combining Spatial Omics Data To Identify More Robust Biomarkers

From literature reviews on biomarkers for hypoxia in HNSCC (see Section 2.3), it becomes evident that many of the proposed unimodal biomarkers prove ineffective for prognosis. Technically speaking, spatial omics itself provides a multimodal view, i.e., spatial and molecular information. However, ideally the findings of biomarker candidates are supported by additional data and sources. For example, consecutive tissue slices might be used to evaluate the abundance of peptide candidates using immunohistochemistry. Alternatively, information from multiple omics layers may be collected, like DNA/mRNA or mRNA/peptides. Of particular biological interest is the relationship of genes and peptides. Proteins perform a critical function in organisms by translating the information encoded in genes. While the pure presence of mRNA levels is not sufficient to predict protein abundance, the existence of proteins implies that a corresponding mRNA was present [160]. Knowing both, gene expression and peptide abundance, might therefore help on the one hand to develop more precise biomarkers. On the other hand, it can add an extra layer of validation, in case some mRNA-protein correlation is encountered.

In this chapter, it is discussed how spatial omics data can be combined. Although only preliminary, I showcase two possible approaches for combination outlined in Sections 5.1 and 5.2. Both strategies rely on what is frequently delivered in course of the spatial experiment itself: H&E stains, typically performed before or after the actual experiment. The chapter concludes by discussing limitations and possible improvements, as well as alternative approaches to combine spatial omics data in Section 5.3.

5.1 Combining Serial Slices from MSI and SPT

Ideally, spatial omics data would be multimodal by design, i.e., providing insights for multiple biological components on the same tissue slice. Although first spatial multi-omics technologies have evolved, there are still some barriers left. For example, for combining spatially resolved genes and proteins, one is currently limited to either certain regions of interest (instead of whole tissue profiling)[26] or to a fixed-sized antibody panels to profile proteins [161, 162]). Alternatively, different spatial omics modalities might be collected from consecutive tissue slices of the same individual and combined computationally. This reflects one of two approaches taken for combining spatial omics modalities in this thesis, visualized in Fig. 5.1. Slice B and C represent direct neighboring tissue slices on which MSI and SPT experiments were performed. Additionally, A (direct neighbor of B) and D (direct neighbor of D) were cut for fluorescence staining with pimonidazole.

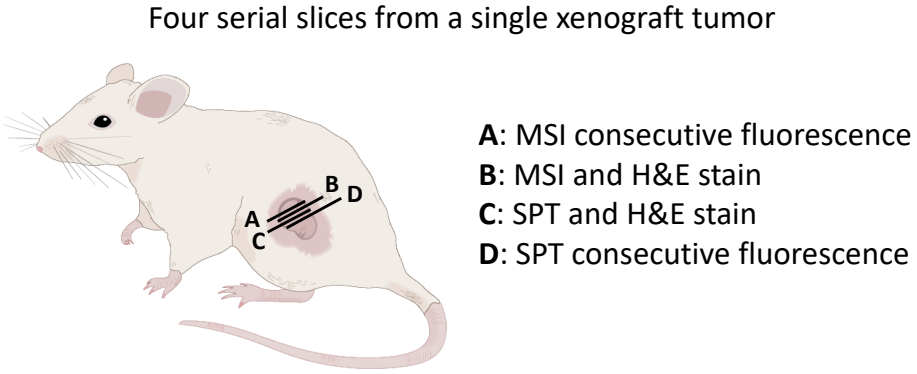


Figure 5.1: Serial tissue slices, i.e., direct neighboring slices, of the same xenograft sample are used for combining mass spectrometry imaging (MSI), slice B, and spatial transcriptomics (SPT), slice C. Further consecutive slices (A and D) were stained with pimonidazole to visualize hypoxic regions. The gaps separating the slices in the sketch are introduced for visualization purposes. I gratefully thank Cristina Conde Lopez for the xenograft illustration.

The described approach comes with a few caveats, summarized in Fig. 5.2. The challenge of high dimensionality remains. More critically, the number of samples is further reduced, given the high complexity in sample preparation: Initially, sample preparation was carried out in two different labs, as MSI experiments were performed at a partner site in Dresden and SPT experiments were carried out at the DKFZ in Heidelberg. However, this lead to

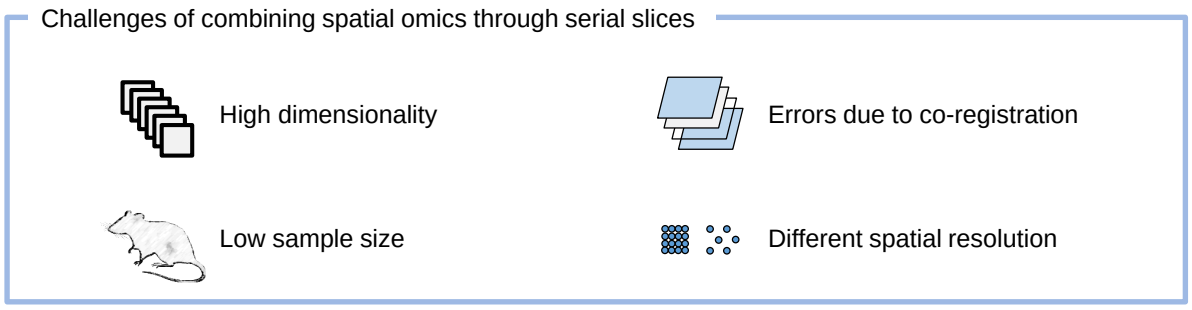


Figure 5.2: Challenges of combining spatial omics through serial slices.

substantial differences in tissue morphology in the allegedly neighbor slices B and C. This was likely caused by different cutting angles, unavoidably when slicing is performed from two persons at two different locations. In the end, the cutting was carried out by one person in one location, leading to the reduced size of just four samples. Another limitation concerns the necessity to combine data through co-registration of different modalities, likely leading to inaccuracies in the process. Lastly, the used MSI and SPT platforms involve different spatial resolutions that must be overcome, with MSI data being organized in a grid-like structure and SPT data in a orange crate manner.

Section 5.1.1 describes the overall workflow for combining MSI and SPT data using serial tissue slices. Following, some preliminary results are presented.

5.1.1 Methods

The slices of each sample were combined independently from other samples. Therefore pre-processing of the underlying data was performed as described in Sections 4.2.1 and 4.3.1, respectively.

5.1.1.1 Co-Registration of Imaging Modalities

Given the inherent difference of MSI and SPT data, like spatial resolution and spot alignment, I decided against a direct co-registration of the data. Instead, a four-step procedure involving the H&E images of the MSI and SPT, was carried out: First, the MSI H&E was aligned to the SPT H&E. Second, an upsampled MSI representative of the data was aligned to the co-registered MSI H&E. Third, the transformations of step 2

without scaling were applied to the actual MSI data. Fourth, potential misalignment due to rounding errors in step 3 was addressed by a grid search to identify the best alignment. In this procedure, SPT was considered to be the *fixed* and MSI to be the *moving* modality. A favorable aspect of this choice is that co-registration of the SPT data and its corresponding H&E was outsourced to the Space Ranger software. Consequently, the geometry of the SPT modality, both data and H&E, remained untouched. For MSI, the corresponding data underwent certain translations and rotations, but the actual size of the spots remained unchanged. Instead, a scaling factor to project the spatial spots to a pre-defined resolution of the H&E image was calculated, approaching the data structure found in SPT data. In the following the individual steps are described in more detail.

Co-Registration of H&E Images

First, the two H&E images were aligned with one another. Low resolution (600 pixels in the largest dimension), grayscale representations were chosen over co-registration of high resolution images on individual channels. First, both H&E images were converted to grayscale and subjected to contrast limited adaptive histogram equalization (CLAHE). Afterwards, pixel values were normalized to a range between $[0, 1]$. Following, the H&E of MSI was aligned to the overall intensity distribution of the SPT H&E by matching histogram characteristics. Then, the same strategy for co-registration was applied as described in Section 4.1.1, i.e., before a similarity transformation was performed (Supplementary Material, Code Snippets 7.1 and 7.2), the MSI H&E underwent an initial alignment to the SPT H&E. Fig. 5.3 shows the two H&E images (top of figure) and two different visualizations of the aligned H&E images (bottom of figure).

Co-Registration of Upsampled MSI Representative

Next, the mean spectra of the MSI data was upsampled to approximate the size of the co-registered MSI H&E. Then, the upsampled representation (moving) was aligned with the MSI H&E image (fixed) using a similarity transformation. From the scaling parameter of the executed transformation, a scaling factor to project spots to the MSI H&E image was calculated. This step is shown in Supplementary Material, Code Snippet 7.13, method *coregister_upsampled()* of class *CoRegistration*.

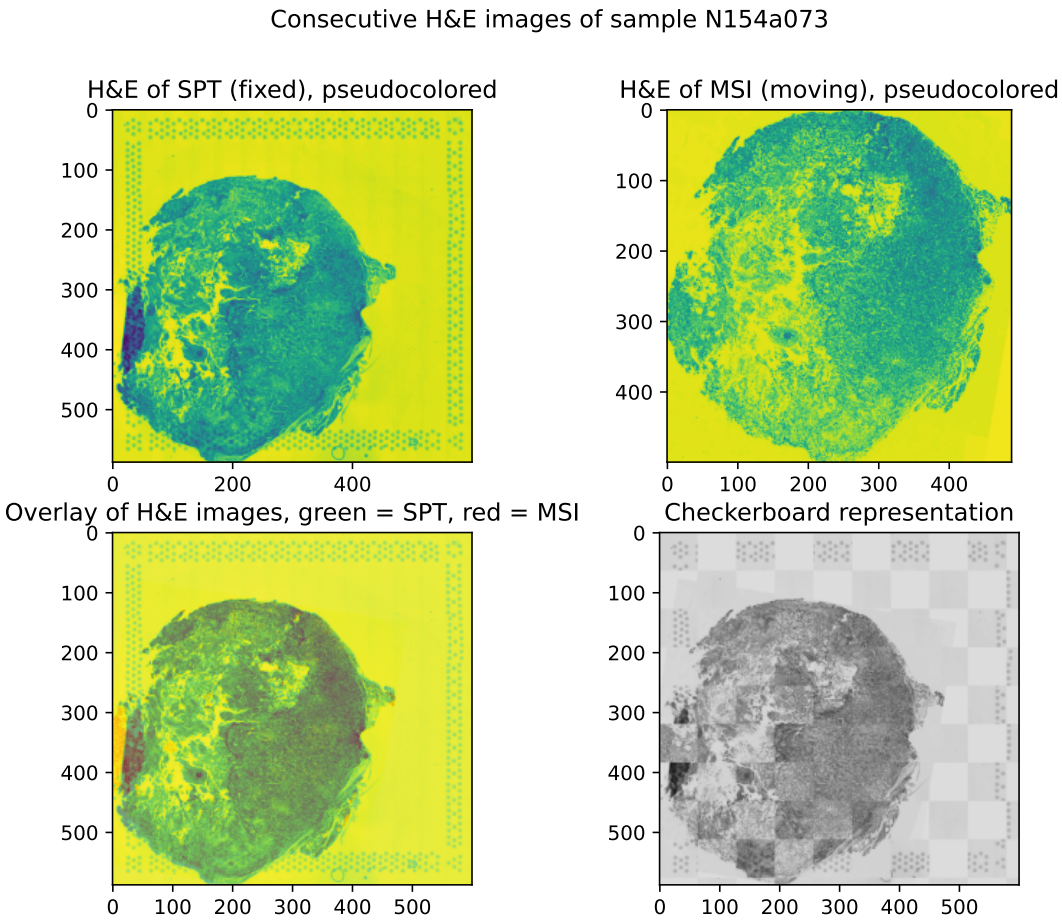


Figure 5.3: Results of co-registration of serial hematoxylin and eosin (H&E) images of sample N154a073. Top left: H&E of spatial transcriptomics (SPT), fixed image, border regions showing fiducial markers. Top right: H&E of mass spectrometry imaging (MSI), initially aligned moving image. Bottom left: Overlay of both HE images, whereas light red and light green represent non-overlapping areas between slices. Bottom right: Checkerboard representation shows discrepancies in contrasts and/or tissue structure especially in the lower left part of the two slices.

Co-Registration of MSI Data

The transformations of the previous step had to be applied the actual MSI data. Typically, transformations in registration software follow a dedicated scheme. For the ITKElastix software, which is based on ITK and elastix, the order of transformation in a SimilarityTransform is scaling, rotation, translation [163, 164]. As the MSI data spots should remain fixed in size, the scaling parameter was set to 1. Thus, the rotation and translation parameters had to be adjusted according to the size difference of the upsampled MSI

representative and MSI data, before applying the transformation to the data. The implementation of this step is shown in Supplementary Material, Code Snippet 7.13, method *apply()* of class *CoRegistration*.

Grid Search for Best Alignment

The results of different transformation parameters were compared in a grid search approach. Therefore, the x and y floating-point translation was adjusted to identify the best setup for the final transformation according to the DICE score of binary representations of the moving and fixed image. This was essential to account for small inaccuracies potentially caused by accumulated rounding errors when transformation was applied to the lower resolved MSI data. The implementation of this step is shown in Supplementary Material, Code Snippet 7.13, method *___find_best_overlay()* of class *CoRegistration*. The final transformation was then utilized to adjust the geometry of all m/z values in the MSI data accordingly (*apply_transform_to_other()* of class *CoRegistration*). Fig. 5.4 shows the low resolution MSI H&E (co-registered to the SPT HE) with the projected spatial spots from the data.

5.1.1.2 Mapping of MSI to SPT Spots

After co-registration, the projected coordinates of the spatial MSI spots to its H&E image were known. The same information was derived for the SPT spots from the output of Space Ranger, as described in Section 3.1.1. Next, the spatial spots of both modalities need to be matched on the projected coordinates. Considering the lower number of spots in SPT, its spots were defined as target. The spots of MSI were then interpolated to the coordinates position of the SPT spots using a radial basis function. As a result, SPT data as well as MSI data per spots became accessible. Following, the Spearman correlation coefficient among genes and peptide information was calculated, to determine which genes and fragmented peptides are co-expressed. Associations with a Spearman correlation coefficient of > 0.5 were considered relevant. Only correlations with a p-value < 0.05 were considered as statistically significant. The implementation of these steps are shown in Supplementary Material, Code Snippet 7.13, methods *interpolate_msi_spots()* and *find_correlating_spots()*, respectively.

A complete example on the individual steps for the combination of serial spatial omics

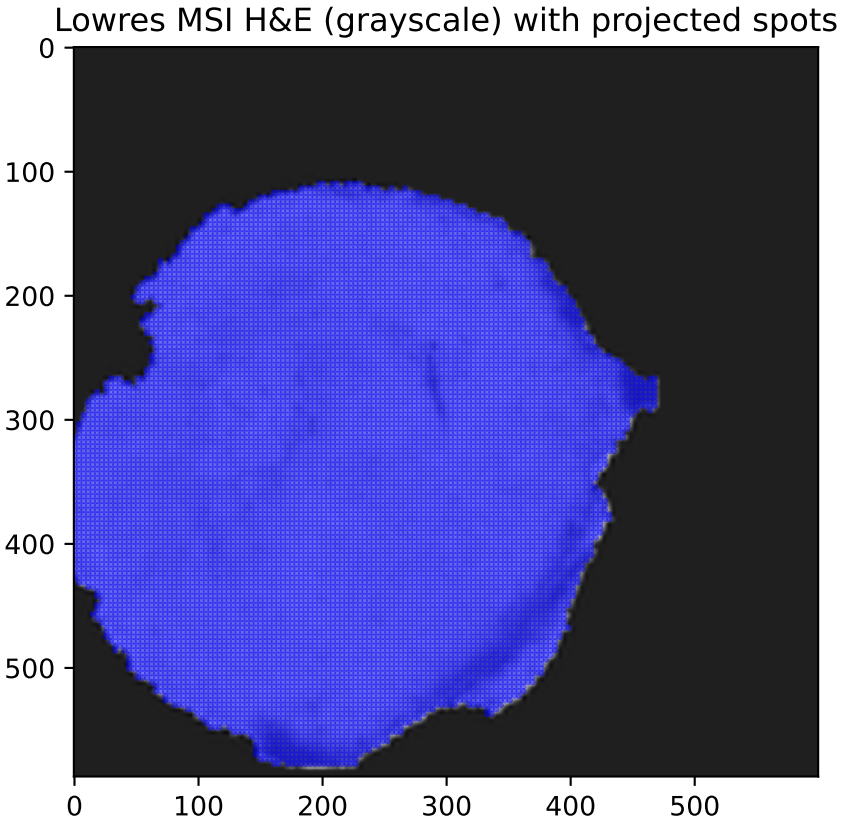


Figure 5.4: Spatial spots projected to mass spectrometry imaging (MSI) low resolution (lowres) hematoxylin and eosin (H&E).

data is demonstrated in Supplementary Material, Code Snippet 7.14, with details of the auxiliary classes shown in Code Snippet 7.15.

5.1.2 Preliminary Results

Table 5.1 shows the results of co-registration of MSI and SPT data for four samples from four tumor models. The shown numbers denote the total number of genes and the total number of MSI masses that were found co-located. Of note, typically one gene (or one mass) was correlated to multiple masses (or genes). For example, *SPRR2D* in N165a002 was associated to 78 (out of 93) masses, whereas its highest correlation was found with mass 628.333. Considering that genes are co-expressed to many other genes, and peptides

Table 5.1: Results from co-registration of serial mass spectrometry imaging (MSI) and spatial transcriptomics (SPT) slices. Numbers denote the total number of correlating genes and peptides with a Spearman correlation coefficient of at least 0.5 within a sample.

Sample	Experiment (MSI/SPT)	Tumor Model	Total number of Genes / MSI mass	Highest correlation Corr coeff.	Highest correlation Gene / MSI mass
N165a002	_M996/S008C	SAT	93/12	0.58	SPRR2D / 628.333
N154a073	_M994/S008A	CAL33	893/80	0.76	KRT16 / 1337.688
N156a074	_M997/S008D	SAS	0/0	/	/
N150d320	_M995/S008B	FaDu	53/2193	0.59	SLC2A1 / 618.323

are co-abundant to other peptides, this pattern was expected to emerge. Among the genes and peptides matched, some were found to be associated with hypoxia in the unimodal approaches of Sections 4.2 and 4.3. For example, KRT16 showed up as protein marker in the CAERF approaches (unsupervised and semi-supervised). Also, when combining MSI and SPT data, KRT16 showed the highest correlation with mass 1337.688 in a CAL33 sample, which is also close to one of the MSI masses (1337.665) used for identifying the peptide in the LC-MS/MS data. Of note, the tissue slices used here, differ from those in the unimodal approaches, thus mass inaccuracies are expected. Fig. 5.5 visualizes

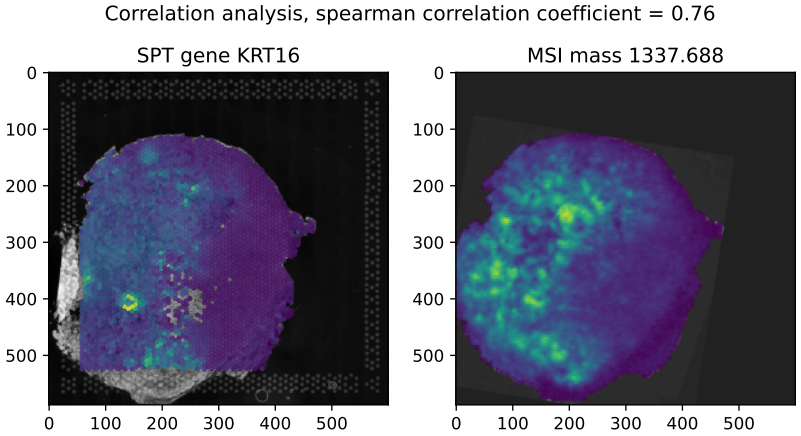


Figure 5.5: Example of gene (left, KRT16, spatial transcriptomics (SPT)) and peptide (right, mass 1337.688, mass spectrometry imaging (MSI)) with a strong correlation (Spearman correlation coefficient of 0.76) in sample N154a073. Shown are the spatial spots on top of the corresponding hematoxylin and eosin (H&E) images. Yellow = high gene expression / peptide abundance, blue = low gene expression / peptide abundance. Areas overlapping with fiducial markers in spatial transcriptomics (SPT) (left) are not covered with spatial spots.

KRT16 and mass 1337.688 in the aligned SPT and MSI data. For the remaining two MSI masses (854.496, 1259.553) associated with KRT16 in the CAERF approaches (854.495, 1259.578), comparable high correlation coefficients of 0.75 and 0.73 were derived. Other genes which map with the findings of MSI include *ANXA1, HSPA8, KRT14, TGM1, LDHA*. Additionally, the genes *KRT16, ANXA1, TGM1, LDHA* were also brought up as associated with hypoxia in the unimodal SPT method. Of note, some promising gene candidates, e.g., *ALDOA, KRT6A, KRT6C, PKM* were excluded in the Visium probset for potential off-target activity, and thus cannot be evaluated.

The only sample for which no strong correlation of genes and peptides was apparent was in SAS sample N156a074. The checkerboard representation of the two H&E images revealed (Fig. 5.6) that the tissue structures displayed significant variations. Reducing the Spearman correlation coefficient from 0.5 to 0.3 resulted in 5 genes and 23 significantly correlated peptides. The only modest associations might be attributed to biological or technical causes such as varying mRNA- protein levels or too diverse neighboring tissue sections, respectively.

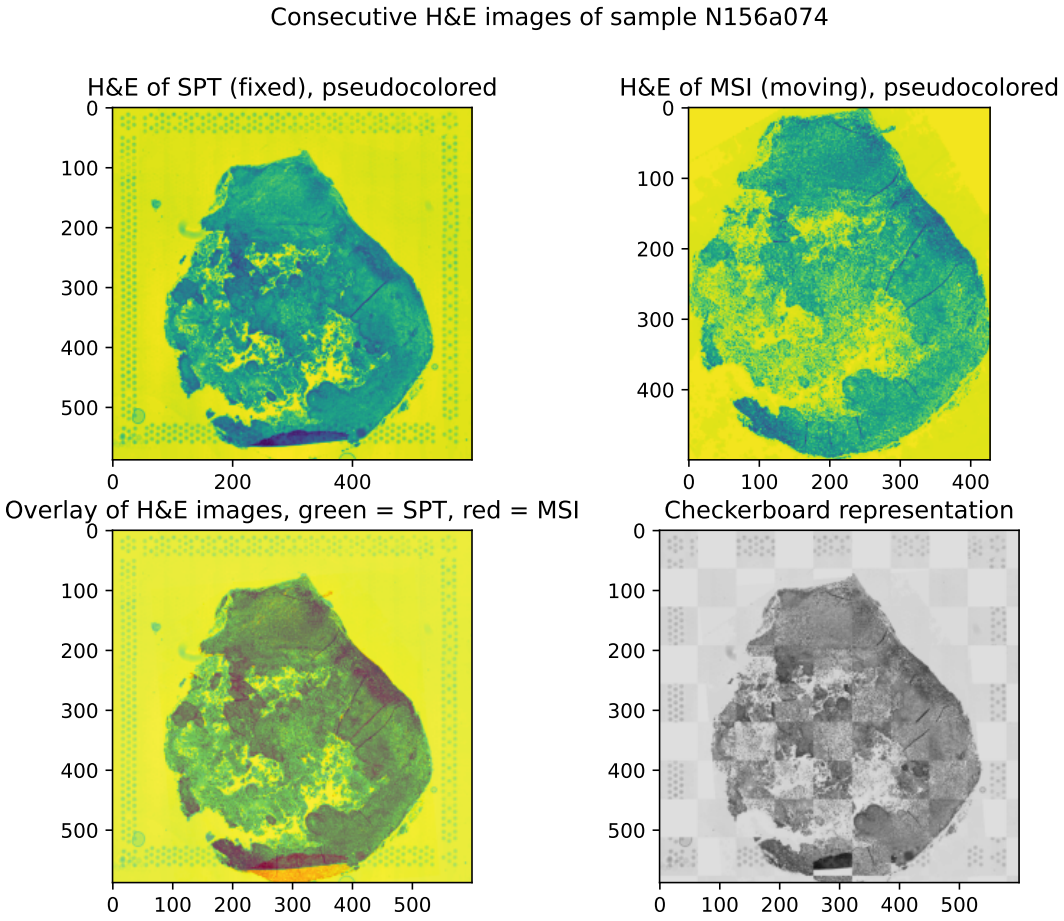


Figure 5.6: Results of co-registration of serial hematoxylin and eosin (H&E) images of sample N156a074. Top left: H&E of spatial transcriptomics (SPT), fixed image, border regions showing fiducial markers. Top right: H&E of mass spectrometry imaging (MSI), initially aligned moving image. Bottom left: Overlay of both HE images, whereas light red and light green represent non-overlapping areas between slices. Bottom right: Checkerboard representation reveals broader discrepancies in contrasts and/or tissue structure.

5.2 Learning Peptide Information from H&E Stains of MSI

Instead of using the H&E stains for combining multiple omics modalities via co-registration, they may be used to learn characteristics of the data itself. The intention is to learn the spatial omics data directly from H&E stains. The idea came up during the discussion with Dr. Denis Schapiro about the inaccuracies of registering consecutive tissue slices. In theory, the trained model would then allow to augment any other H&E stain from the same tumor entity to predict the spatial omics data on top of it. If applicable, this strategy would reduce the necessity for executing time-consuming spatial omics experiments once the model is sufficiently trained. Considering that H&E stains are routinely acquired in clinical practice for cancer diagnosis and grading [42], augmenting these images with molecular information would also facilitate the adoption of new biomarkers substantially. On a more specific note for the combination of spatial omics modalities, it would allow to reduce the number of consecutive slices that require co-registration.

In this Section, it is evaluated whether it is possible to learn MSI derived peptide information from H&E stains using a CNN. In particular, it is investigated whether hypoxia-associated peptides can be learned from H&E images. While it would also be conceivable to train the CNN on SPT data instead of MSI data, a lot more MSI data is available from the tumor models under investigation. The subsequent steps of predicting the MSI data on H&E images from SPT experiments are not covered in this thesis, but will be incorporated as part of future work. Fig. 5.7 sketches the main challenges that

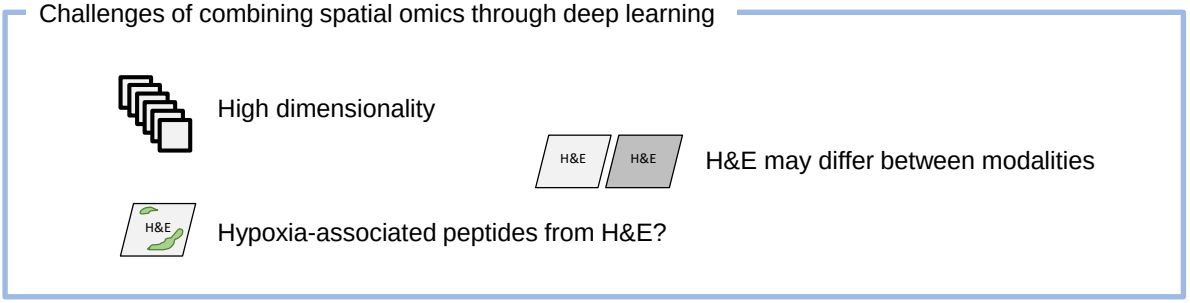


Figure 5.7: Challenges of combining spatial omics through deep learning approaches.

need to be addressed. First, high-dimensional spatial omics data is typically sparse, making it challenging to unravel biological patterns. However, the strong correlation of features might facilitate the learning process. Second, the H&E stains utilized during training might look very different to the H&E stains on which prediction is performed. Consequently, the training data needs to be augmented accordingly. Third, while it might be possible to learn some molecular candidates from H&E stains, in course of this thesis it is of particular interest whether hypoxia-associated peptides can be learned.

Two strategies are proposed to learn peptide information from MSI H&E images: In the first one, a CNN is trained to predict pre-processed MSI data from H&E patches. In the second one, a CNN is trained to learn the reduced latent space representation of the proposed CAE architecture described in Section 4.2. In the following, Section 5.2.1 describes the data and workflow of the two methodologies. Some preliminary results are shown in Section 5.2.2.

5.2.1 Methods

Pre-processing of MSI data was performed as described in Section 4.2.1. This section focuses on the co-registration of the H&E image to the data and corresponding creation of patches.

5.2.1.1 Co-Registration of H&E Images

For training a CNN, the H&E images and the MSI data were split into a pre-defined number of patches. This strategy allows to reference both, image and data patches, by the same index. Therefore, image and data had to be co-registered accordingly. The MSI data, i.e., given the difference in dimensions, an upscaled representative of the MSI data, was considered to be fixed. The H&E image was set as the corresponding moving image.

Different from the low resolution images used for the co-registration of serial tissue slices (Section 5.1), two different H&E resolutions were utilized to prepare the images for the deep learning approach. First, the original H&E image was exported using QuPath with a downsampling factor of two. Therefore, the Open Microscopy Environment TIFF (OME-TIFF) file format was employed, to store different levels, so-called image pyramids. A level 2 image pyramid (up to 1,500 pixels in its largest dimension) was used for initial alignment

with the fixed image (auxiliary class *MovingDummyChannel* in Supplementary Material, Code Snippet 7.16). As it is important to preserve the individual color channel information for the model, the registration was performed on the channels individually, starting with the red channel. Therefore, the red channel image was cropped while retaining the aspect ratio to the data. Following, a mean spectra image as representative of the MSI data was upsampled to match the size of the cropped moving image. Considering that the channels of an H&E image contain many local structures in contrast to the data, a binary mask of the red channel was co-registered to a binary mask of the MSI representative. For co-registration, a similarity transformation was applied, optimizing the mattes MI metric. Then, a level 0 image pyramid (up to 19,000 pixels in its largest dimension) of the H&E image was loaded (auxiliary class *MovingRGBImage* in Supplementary Material, Code Snippet 7.17). Next, the co-registered level 2 red channel image was upsampled to approximately match the size of the level 0 image. The level 0 image (all channels) was cropped relatively to the level 2 image, accounted for the size difference. Subsequently, the registration of the level 0 to the level 2 representation by means of a similarity transform and mattes MI metric was executed. More precisely, the level 0 red channel was used for registration with the level 2 red channel, and the resulting transformation was applied to the blue and green channel accordingly.

In some final steps, the co-registered level 0 image and the underlying data were cut into patches. The number of patches depends on a sample's data dimensions. Considering the spatial data spots to be fixed in dimensions, the dimensions of the level 0 image needs to be adjusted accordingly. For the dimensions of some given data, a pre-defined data patch size and a pre-defined image patch size, the dimensions of the H&E image can be calculated. For example, assuming the dimensions of the data to be 129×129 pixels, a data patch size of 3×3 pixels and a image patch size of 360×360 pixels, then, the corresponding H&E image should be the 120 fold of the dimensions of the data, i.e., 15480×15480 pixels in size to retain the aspect ratio. The image patch size should be a multiple of the data patch size to facilitate the generation of overlapping patches. For the given example, a shift by 1 pixel in the data would correspond to a shift by 120 pixels in the image. The image's dimensions should not exceed its actual size to avoid upsampling of the image. The number of patches can be calculated thereafter with the following

formula:

$$\text{number of patches} = \frac{\text{dimension}^2}{\text{patch size}^2}$$

In the used samples, the data patch size was set to 3×3 pixels and the image patch size to 360×360 pixels. In case the dimensions of the data were asymmetric, they were padded accordingly. The level 0 H&E image was downsampled to match the computed image dimensions, taking into account the padding of the data. For example, considering the dimensions of MSI data to be 124×128 pixels, it is padded by 5×1 pixels to 129×129 pixels. As the image patch size is 120 times larger, the padding of H&E image is expected to be 600×120 pixels. Thus, the H&E image is downsampled to match the computed dimensions minus 600×120 pixels. As a final step, the H&E image is padded accordingly. Given the limitation to whole numbers, some alignment errors might be introduced due to rounding errors in the progress. Assuming that the actual image is downsampled to 14885×15360 pixels (instead of 14880×15360 pixels), preserving approximately the aspect ratio of the data, this would lead to a padding error of -5×0 pixels. For the samples presented, the maximum error accounted was 0×-7 pixels, presumably too low to have a significant effect on learning. The precise implementation is shown in method *derive_params()* of class *ImagePatchBuilder*, Supplementary Material, Code Snippet 7.18.

To further mitigate the effect of padding errors or inaccuracies during co-registration, overlapping patches were created with a step size of 1 data pixel and 120 image pixels accordingly. A complete example for the co-registration of images and patch creation can be found in Supplementary Material, Code Snippet 7.19.

5.2.1.2 Strategies for CNN Learning

Two different learning strategies are proposed.

In the first approach, the abundance pattern of the pre-processed 18,735 m/z values should be acquired by a CNN. Therefore, the data patches were reduced from $3 \times 3 \times 18735$ to 1×18735 by computing the mean intensity value per patch. Then, the H&E patches were inputted together with the intensity values to a CNN.

In the second approach, the CNN should learn latent m/z values from the latent space of a previously trained CAE. Therefore, the data patches were encoded using the proposed unsupervised CAERF approach, based on the weights of the presented qualitative run

(Section 4.2.2.1). In this case, the data patches were reduced from $3 \times 3 \times 18735$ to $2 \times 2 \times 64$. Again, the mean intensity value was computed per patch, resulting in a final vector of 1×64. The encoded values were normalized to a range between $[0, 1]$. The H&E patches and the reduced data patches were then fed into the CNN.

5.2.1.3 Sample Size

For demonstration purposes, four out of five CAL33 samples which were used to train the CAERF approach were utilized. Fig. 5.8 shows exemplary the aligned H&E and

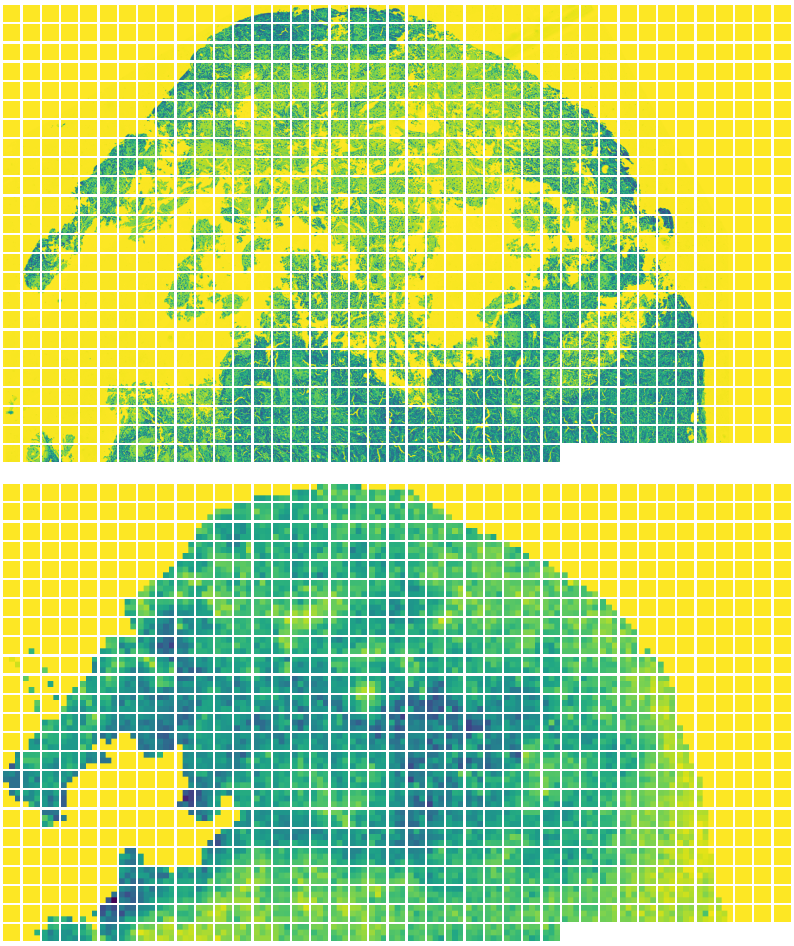


Figure 5.8: Patches of sample _M819_N154a037 (pseudo-colored). Top: Patches from hematoxylin and eosin (H&E)of size 360×360 pixels. Bottom: Patches from mass spectrometry imaging (MSI) data of size 3×3 pixels of exemplary mass-to-charge ratio (m/z) value. In total 972 patches out of 1681 patches are shown. Holes in H&E image likely show areas of previously necrotic tissue or disruption from the MSI experiment.

data patches of one sample. It also highlights the inherent difference in structures between modalities. In H&E images, holes within tissue areas are clearly visible, which is potentially caused by a combination of previously necrotic tissue and disruption from the MSI experiment itself. For the experiments, the tissue was covered with a matrix, which depicts as technical signals in the data. Consequently, the data does not show signs of disruption, even when no actual tissue but only matrix was measured.

Patches showing only background in either the data or the H&E were removed from further examination. The patches for training were further restricted to balance the number of hypoxic and normoxic patches. If the m/z values (first approach) or latent m/z values (second approach) of interest are primarily present in hypoxic patches, than a model would hardly learn linked visual patterns in case they are underrepresented. For three samples, hypoxia labels were available. For the fourth samples, the amount of hypoxia was considered too low. All hypoxic patches with a minimum degree of hypoxia ($threshold > 0.1$) were considered hypoxic, resulting in 4,443 hypoxic patches. Additionally, 6,443 normoxic patches were sampled from the patches not fulfilling the hypoxia threshold. The same amount of hypoxic and normoxic patches were derived from each sample, despite sample $\_M821\_N154a098$, from which only 2,000 normoxic patches were drawn. Hypoxic and normoxic patches were independently split with a 80/20% ratio into training and test sets. The patches were shuffled for training.

5.2.1.4 CNN Training

In the following, italic names denote the corresponding parameters in tensorflow (see Section Software).

For training, a DenseNet architecture with 121 layers was configured (*DenseNet121*). An input shape of $360 \times 360 \times 3$ was set to fit the defined image patch size. No pre-trained weights of ImageNet were used as this would require a patches of size $240 \times 240 \times 3$. Instead of a classification task, a regression task need to be performed to predict the intensity values of m/z values. Therefore, the model's layers were extended by a global average pooling layer (*GlobalAveragePooling2D*) a normalization layer (*BatchNormalization*), and a dense layer (*Dense*) with either 18,735 or 64 output nodes and a sigmoid action function. The adjustment of the DenseNet architecture is illustrated in Code Snippet 5.1. For both approaches, optimization was achieved by a Huber loss function with delta being set to

Code Snippet 5.1: Adjustment of DenseNet architecture for learning peptide information

```
import tensorflow as tf
net = tf.keras.applications.DenseNet121(
            include_top=False,
            input_shape=(360,360,3),
            weights=None,
            input_tensor=None,
            pooling=None,
            classifier_activation=None,
        )
out = tf.keras.layers.GlobalAveragePooling2D()(net.output)
out = tf.keras.layers.BatchNormalization()(out)
output_layer = tf.keras.layers.Dense(n_features, activation='sigmoid')(out)
model = tf.keras.Model(inputs=net.input, outputs=output_layer)
```

1.35 to account for potential outliers especially in the non-reduced MSI data. The training patches were augmented by a factor of eight, applying randomly geometric transformations (rotation, flipping) and color adjustments (saturation, brightness, contrasts, hue). The test patches remained unaltered. Both approaches run with a batch size of 32 and for 150 epochs, with no essential improvements being made after epoch 135.

5.2.2 Preliminary Results

After training the CNN, correlation analyses of the actual and predicted values were performed. Fig. 5.9 shows the Spearman correlation coefficients between learned and actual intensity values in the test set of both approaches. With a p-value < 0.05, all correlations were found to be statistically significant. As expected, the approach on the entire set of m/z values showed overall lower correlations compared to the reduced latent feature set. Next, the features with the lowest and highest Spearman correlation coefficient were examined. For the latent feature approach, the highest correlation (0.65) was achieved for a feature (latent id #32, Fig. 5.10, left) which mainly differentiates mouse and tumor areas according to the corresponding H&E stain. Considering that these differences are also clearly visible by eye, it seems plausible that a CNN can learn the differences. The latent feature (latent id #44, Fig. 5.10, right) with the lowest correlation (0.03) appears to depict technical signals of the matrix and some stroma tissue of the mouse. The majority of patches showing matrix signals were dismissed during CNN training as the corresponding H&E patches typically show no tissue. The absence of some

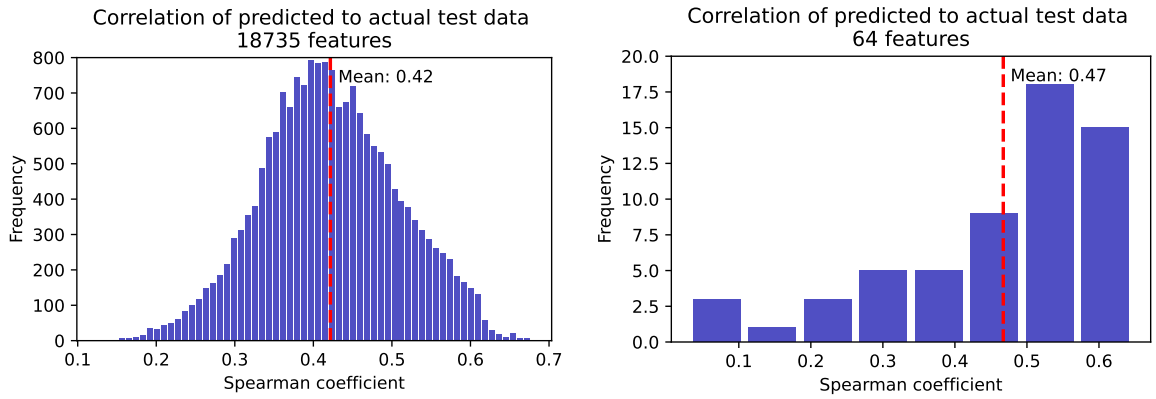


Figure 5.9: Correlation analysis of predicted and actual peptide information. Left: Results of convolutional neural network (CNN) trained on 18,735 mass-to-charge ratio (m/z) values. Right: Results of CNN trained on 64 latent features of the unsupervised convolutional autoencoder (CAE) approach.

molecular features during training may explain the low correlation coefficients in Fig. 5.9. M/z value 775.666 illustrates rather noisy signals (Fig. 5.11, right), with some signal at the border of the tissue slices, and was the feature with the lowest correlation coefficient (0.13) in the CNN approach, trained on the raw m/z values. The m/z value 701.383 with the highest correlation coefficient (0.68) depict tumor regions (Fig. 5.11, left).

For comparison, the H&E images are shown in Fig. 5.12, where pink areas show mouse tissue and purple areas belong to tumor cells of human HNSCC.

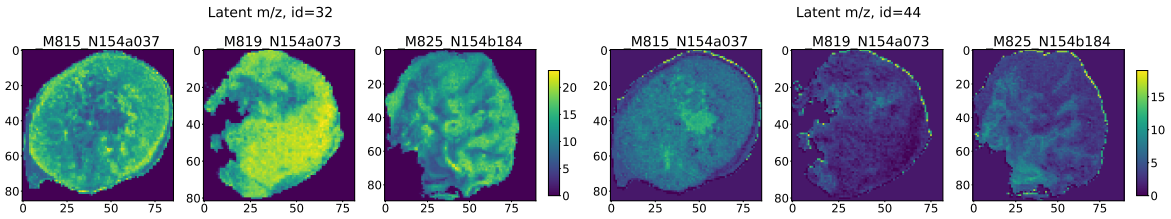


Figure 5.10: Exemplary images of latent features with highest (left, #32, 0.65) and lowest (right, #44, 0.03) Spearman correlation coefficient of predicted and actual peptide intensity values.

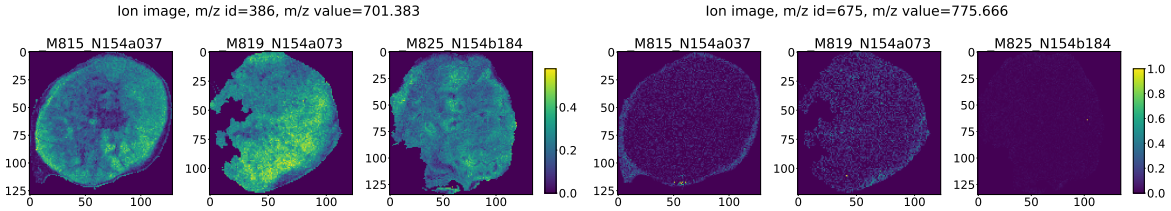


Figure 5.11: Exemplary ion images of mass-to-charge ratio (m/z) values with highest (left, 701.383, 0.68) and lowest (right, 775.666, 0.13) Spearman correlation coefficient of predicted and actual peptide intensity values.

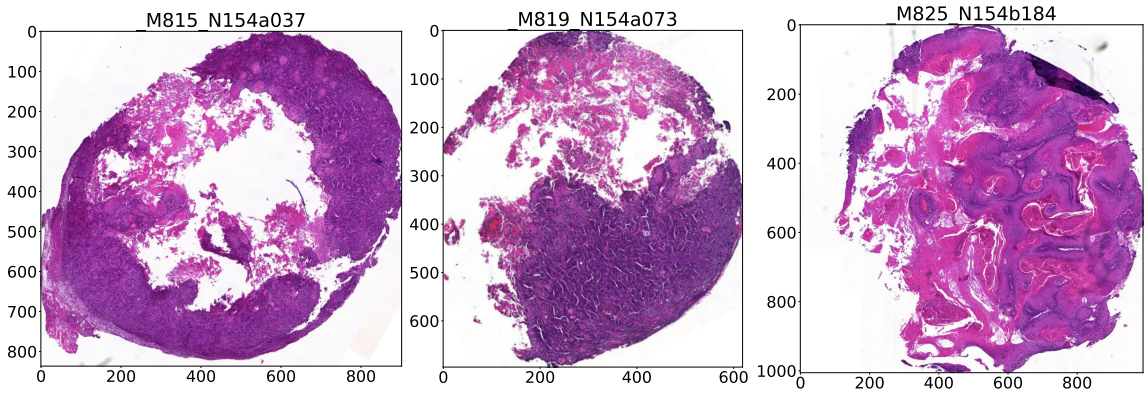


Figure 5.12: Corresponding hematoxylin and eosin (H&E) images from 3 out of 4 samples from which the convolutional neural network (CNN) learned peptide information. Pink areas denote mouse tissue like stroma, purple areas show tumor tissue.

However, the overall performance is only of secondary interest. Instead, the main focus is directed towards uncovering evidence, whether it is possible to acquire patterns of features associated with hypoxia. Therefore, m/z values which found to be associated with hypoxia according to the results of Section 4.2.2 were investigated.

Table 5.2 shows some exemplary raw m/z values and latent feature 56, which were found to correlate with hypoxia annotations, as well as their actual and predicted value. For the raw m/z values, the mean difference between expression values in hypoxic and normoxic patches is relatively small and the median absolute error relatively high. Therefore, it remains unclear, whether the CNN can truly differentiate normoxic and hypoxic patches for the shown features, as it underestimated expression values particularly in the hypoxic values. In contrast, the expression values for the actual and predicted latent feature appeared resistant to mix up. With this first attempt to learn MSI data from H&E stains, the results yield promising that some peptide information, particular those that might be linked to tumor hypoxia, can be connected to visual patterns.

Table 5.2: Hypoxia-associated features and their expression in hypoxic and normoxic patches in the test set. The first five rows show raw m/z values, the last one the latent feature from the convolutional autoencoder (CAE) approach. The shown overall correlation coefficient denote the Spearman correlation coefficient of predicted and actual intensity values for the given m/z value. Column y corresponds the actual mean intensity values, y_pred corresponds to the mean predicted values, and $MedAE$ denotes the median absolute error between actual and predicted values for a specific feature.

Metrics of prediction of hypoxia-associated features

		Overall	Normoxic Patches			Hypoxic Patches		
m/z id	m/z value	corr. coeff.	$\bar{y}\_pred$	$\bar{y}$	MedAE	$\bar{y}\_pred$	$\bar{y}$	MedAE
1548	989.48	0.59	0.16	0.19	0.08	0.21	0.27	0.12
1583	998.47	0.55	0.19	0.20	0.11	0.22	0.26	0.10
1587	999.48	0.44	0.26	0.26	0.12	0.31	0.35	0.10
2382	1167.61	0.55	0.21	0.21	0.11	0.25	0.26	0.10
1785	1044.53	0.65	0.17	0.20	0.08	0.22	0.27	0.08
56	latent	0.62	0.43	0.45	0.08	0.61	0.57	0.08

5.3 Discussion

Two strategies for combining spatial omics data were proposed. In the first one, co-located genes and peptides were derived by combing the spatially resolved molecular information from two consecutive tumor tissue slices. In the second strategy, it was investigated whether it is possible to learn hypoxia-associated peptide information from H&E stains of the same tissue slice to overcome inaccuracies from consecutive slices. Strengths and weaknesses of the strategies are discussed in the following. In the later part of this section, alternative approaches are reviewed.

In the consecutive slice approach, several genes and peptides, which were associated with hypoxia independently from each other in the unimodal approaches, were found to co-locate. These gene-peptide candidates show potential as biomarkers for tumor hypoxia. From a technical viewpoint, the co-registration of neighboring slices is likely to introduce some inaccuracies. Although different layers were co-registered, the most susceptible to errors is the one where H&E images from different slices are aligned. Considering that two consecutive slices span a range of at least $7\mu m$, it is unlikely that the same cells will be consistently observable across different images. Therefore, co-registration may be guided by more global structures e.g., tissue morphology. Hence, I decided for lower resolved images (600 pixels in the largest dimension) and utilized grayscale representations instead of individual channels for alignment. In order to conserve finer tissue structures, the use of high resolution images may be considered as potential future improvement. As a first attempt, only similarity transformation was applied. However, as pointed out in the unimodal sections, some degree of deformation might be introduced to the tissue slices during sample preparation. These local deformations cannot be represented by solely linear transformations. Although I experimented with some additional b-spline transformations for the registration of the H&E images, the results did not significantly alter. Potentially, the varying thickness ($\sim 5\mu m$ for SPT versus $\sim 2\mu m$ for MSI) and a certain degree of tissue disruption caused by the MSI experiments may impede the process. The unaligned H&E images in Fig. 5.3 support the assumption that tissue slices of MSI expose a less dense tissue structure. The figure also shows that contours of the tumor slices do align after co-registration, apart from some overlapping tissue in the lower left corner of the SPT H&E.

Limitations of the consecutive slice approach might be overcome with the proposed CNN approach. Although no multimodal biomarkers were derived in this case by now, the preliminary results suggest that H&E stains may be utilized to learn peptide information. When dimensionality reduction was carried out prior to learning, stronger correlations between the predicted and actual intensity values could be achieved than without it, indicating that the approach had some denoising effects. Surprisingly, also the approach on the entire set of m/z values delivered moderate correlations. However, the prediction error was presumably too high to clearly differentiate hypoxic and normoxic patches for the investigated features. In contrast, the comparison of the predicted and actual intensity values for latent feature #56 associated with hypoxia suggested that the CNN was able to identify some visual distinctions between the labelled patches.

Despite the potential advantages of the CNN approach over the serial slice approach, one particular drawback of the former is that the trained model will perform well presumably only on H&E images from HNSCC xenograft models. Especially structures which differ in purely human tissue, like human immune cells or vessels, are likely to be misperceived. However, considering the large amount of H&E stains and MSI available for the well-characterized HNSCC models, they can be still leveraged for biomarker discovery. It is important to note, that so far the association of the m/z values to hypoxia was inferred only indirectly from findings of Section 4.2.2. In the proposed CNN method, only the molecular and staining information from the same tissue slice, but not the hypoxia labels from neighboring slices were utilized during training. Hence, whether the peptides are actually contributing to hypoxia necessitates further validation. A weak indication would be the direct inference of hypoxia from H&E images. To achieve this, one could attempt to train a deep learning model to predict the level of hypoxia, e.g., between 0 and 1. A high spatial concordance of patches that were predicted to be highly hypoxic by one model and likewise exhibited a high abundance of presumably hypoxia-associated m/z values by my proposed model, would point towards an actual association. Still, this can only be considered as weak indication, since labelled hypoxic regions may be confounded by other, more easily observable tumor characteristics such as metastastic cells. Moreover, there might be several reasons why the prediction of hypoxia from H&E images alone may fail. First, FIs and H&E images represent two independent tissue slices, where hypoxic cells which may be present in the FI slice, may not exist on the H&E slice and vice

versa. Second, it might not be possible to predict hypoxia solely on visual structures. So far, there is only limited evidence that hypoxia can be estimated from H&E images. Sundstrom et al. tried to derive hypoxia-associated features from H&E by means of measuring gradients near blood vessels and employing multithresholding to segment tissue into normal, hypoxic and necrotic regions [165]. However, they acknowledged that the nature of hypoxia involves inherent complexity and that the features proposed can only capture a small proportion of it. Manescu et al. used a multiple instance learning CNN to predict hypoxia from H&E stains, based on the stratification of entire samples into being either hypoxic or normoxic [166]. The stratification was applied according to a hypoxia gene signature. When analyzing the models performance, they detected differences in the morphology of macrophage cells between hypoxic and normoxic samples. However, whether this distinction is a result of hypoxia or other tumor characteristics is challenging to discern. Overall this suggests that a validation by means of additional data may be more sound. Nonetheless, the inability to infer hypoxia from H&E images would not exclude the potential to learn hypoxia-associated peptide information from H&E images. It might be, that only a combination of tissue structure and molecular information enable to unravel characteristics of hypoxia.

For other tumor characteristics, CNNs displayed encouraging findings in the identification of biomarkers from H&E stains [167]. For example, Kather et al. showed that microsatellite instability can be predicted from H&E images, allowing to infer whether patients with gastrointestinal cancer could benefit from immunotherapy [168]. Also, the combination of H&E images and spatial omics data, particularly on SPT data, has been explored previously. Monjo et al. evaluated if a CNN can be employed to impute realistic expression values of SPT data on consecutive slices [169]. Therefore, the model was trained on sections D1 and D3 for imputing expression values in section D2. Among other metrics discussed, they achieved a mean Pearson correlation coefficient in the 23 genes of 0.369 to 0.458 in different models. Additionally, they investigated the performance of different models on 18,542 all genes, compared to 21 or 3 selected genes. Also He et al. used a DenseNet121 to learn 250 genes from 23 patients with breast cancer from SPT data [170]. Validated on an independent dataset, a mean Pearson correlation coefficient of 0.33 across 233 genes was achieved. In a review conducted by Schneider et al., they found that studies combining spatial omics data and H&E images are carried out mostly on a small number of individuals [171]. They concluded that due to the high spatial resolution, insights

into tumor heterogeneity might be still feasible. However, they also emphasized that more external validations are imperative. The only modest correlations in the mentioned papers suggest that raw gene expression data in individual samples may be too unstable for reliable learning and prediction. I hypothesize that extracted latent features from autoencoders are likely to generalize better to unseen data.

Both proposed combinational approaches are limited to mRNA and peptides that co-locate. Although typically some correlation of genes and peptides exists, the strength might vary considerably (Buccitelli et al. [160]). Typically, only modest within-gene correlation can be found, i.e., according to Buccitelli et al. "how much a change in mRNA level of one gene can explain changes in protein levels". This can be partially ascribed to the fact, that the half-lives of proteins are longer than for mRNA, making it likely to detect proteins but miss corresponding mRNA transcripts. As further outlined by Buccitelli et al., the spatial correlation might also be affected by proteins that are transported to another location where they had been produced. Overall, results from the approach on consecutive slices allowed to identify moderate correlations (Spearman correlation coefficient of at least 0.5) in 3 out of 4 samples. Another aspect influencing both approaches is the difference in spatial resolution of the omics modalities. While the data of MSI is structured as grid, the data of SPT is arranged in an offset pattern, resulting in less spatial spots. Adding to this point, the exact spot size in diameter differs ($50\mu m$ in MSI versus $55\mu m$ in SPT). Currently, the approach on consecutive slices is limited by the lower number of spatial spots in the SPT data. By combining the information of the H&E stain with the SPT data, it might be possible to predict higher resolutions [169, 172]. The same principle was applied to MSI data previously [173]. A higher resolution would enable to consider more spots with less interpolation, expecting to improve the proposed correlation analysis in the process. The same considerations hold true when actual SPT and predicted MSI values from a CNN are combined.

Several other measures for improvements are under consideration. For the approach on serial slices, a significant upscaling in sample size is challenging. The main challenge arises from the amount of manual work necessary to ultimately derive biomarkers, especially for sample collection. Nevertheless, 16 more samples are currently processed to allow for creating more stable results of individual tumor models. Even with an increased sample size, the high amount of feature comparisons make false positive associations statistically

likely. Instead of combining all gene and peptide information, a combination of the latent space results from the autoencoder approaches is intended. In particular in case of the MSI data processed here, it may be argued that aggregated peptide information could yield stronger associations with genes than individual m/z values do. Of additional benefit, the comparisons would be restricted to a few latent features of interest, thereby reducing the number of false positive correlations.

For the CNN approach, it is important to train the model on more data for better generalization. Additionally, data augmentation is currently limited to basic geometric transformations and color adjustments. However, from observations of H&E images of SPT, it became apparent that the thicker slices result in a more densely packed tissue structure. Therefore, it needs to be investigated if specific data augmentation is indicated and feasible. Also, it might be worth experimenting with a different parameter setup. Currently, the image patch size of 360×360 pixels was chosen in a way that the level 0 H&E image required only minimal additional downsampling. The data patch size of 3×3 pixels was selected to mitigate outliers in individual pixels and to allow for dimensionality reduction with the trained autoencoder. It might be, that a different set of data patch size and image patch size yields better results. Especially transfer learning by using pre-trained weights for the CNN might improve performance further. As the DenseNet implementations typically expect a input size of 224×224 pixels, either the architecture itself or the patch size would need to be adjusted. For the CNN approach on the entire feature set, the proposed adjustments to the architecture may be only suboptimal, considering that the dimensions of the last layer before the introduced pooling layer are (11, 11, 1024). A subsequent pooling layer will reduce the dimensions down to 1024, which might be too low to predict 18,735 features. However, also additional upsampling (e.g., by a subsequent dense layer with 4,096 nodes) did not improve results. In general, previously published work on deep learning strategies for H&E images showed that a different architecture itself may alter results [169]. Therefore, also the performance of other architectures with fewer convolutional layers, like VGG16, might be worth exploring.

Some of the findings, especially the multimodal biomarkers of the approach on consecutive slices, already allow for an independent validation. As initial step, the correlation of the markers to the consecutive pimonidazole-stained slices A and D (Fig. 5.1) will be evaluated. Ideally an expert, preferably unaware of the findings, would annotate the FI

accordingly. Alternatively, the spatial pattern of peptide candidates might be compared against further consecutive slices stained for antibodies via immunohistochemistry. The same strategy can be applied to evaluate the visual structures that are picked up by the CNN to learn (hypoxia-associated) peptide information. This requires the employment of explainability techniques for the learned CNN model, like filter visualization or utilizing integrated gradients [170]. Once prediction is applied on SPT images, true neighboring MSI and SPT tissue slices might be beneficial for further evaluation: Given a MSI slice B and a SPT slice C, the predicted MSI values on slice C can be rated by the actual measured MSI values of slice B. Assuming that neighboring slices share a high degree of visual structures and molecular information, similar expression values should be predicted.

While the integration of multi-omics data encompasses a wide spectrum of research, so far the combination primarily focused on non-spatial omics data [174]. Certainly, many of the proposed strategies can be adapted to incorporate spatial information. For example, one obvious extension of the proposed CAE approaches would include the fusion of latent space features by another autoencoder, as proposed by Xu et al. [175]. Alternatively, an early integration of data by means of a single autoencoder might be implemented [176]. However, in both cases the integrating autoencoder would be unaware of the fact that data from the same individuals are being processed. Instead, dedicated pairs (e.g., from individuals) might be passed to the autoencoder to allow for contrastive learning. In contrastive learning, the loss function is adapted to maximize the similarity between positive pairs and minimizing the similarity of negative pairs [177]. Zhou et al. were among the first who proposed contrastive learning for multi-omics integration by maximizing the MI between different omics layers [178]. Given many more paired samples can be used in a patch-wise spatial omics approach different from non-spatial omics data, this integration approach appears promising. Other techniques include graph-based integration, in which either relations within one domain (e.g., protein-protein interaction) or between domains (e.g., multiple omics) are depicted [179]. A specific form of it represent graph neural networks. In principle, it is possible to combine various concepts for deep learning integration. For example, Rajadhyaksha et al. proposed graph contrast learning for multi-omics integration by constructing omics data as graphs and utilizing contrastive learning as pre-training strategy [180]. The persisting challenge, however, remains the topic of explainability in deep learning approaches. In the unimodal

approaches proposed in this thesis, the recovery method incorporated correlation analysis to identify contributing original features to a latent feature of interest. However, by adding additional models (e.g., an integrative autoencoder) or concepts (like graphs) more elaborate strategies are likely to be required for unraveling the added complexity.

In summary, several multimodal biomarkers for hypoxia were identified which are in line with findings from unimodal approaches. The preliminary findings suggest that CNNs are capable of identifying relevant patterns to predict peptide information, especially hypoxia-associated peptides. This could significantly ease the combination of spatial omics modalities by reducing the amount of consecutive tissue slices in the long run. Errors caused by the use of consecutive slices and subsequent inaccuracies from co-registration may arguably be eliminated. To the best of my knowledge, no comparable strategies for combining spatial omics modalities have been published so far, particularly in combination with dimensionality-reduced data from CAEs.

6 Conclusion

This thesis aimed to enhance biomarker discovery for tumor hypoxia in HNSCC by means of advancing accessibility of spatial omics data through convolutional autoencoders. Exemplified on data of SPT and MSI, it was shown that convolutional autoencoders enable to extract also features which exhibit only modest signals like in the case of tumor hypoxia. The thesis highlights key aspects on how to build an autoencoder architecture for spatial omics data that allows for explainable feature extraction: Given a latent feature of interest, the corresponding recovery method estimates the contribution of all original features to it via correlation analysis. The specifics of individual spatial omics modalities (like the orange crate packing in SPT), have been addressed effectively through the adjustments of the autoencoder's hyperparameters (e.g. kernel size) and its loss function (weighting). It was demonstrated that the combined convolutional autoencoder and random forest (CAERF) approaches, identified less noisy associations to tumor hypoxia than random forest models alone. In particular, the proposed semi-supervised convolutional autoencoders, which incorporated hypoxia annotations during training, reduced noisy associations further as indicated by the significantly higher structural similarity index measure scores.

Several unimodal biomarker candidates for hypoxia in MSI and SPT data were identified. Some of these peptide or gene candidates were associated already previously with hypoxia (like Phosphoglycerate kinase 1 *PGK1*, Pyruvate kinase *PKM*, L-lactate dehydrogenase A chain *LDHA*). Other candidates have not been previously known to contribute to hypoxia (like Keratin, type I cytoskeletal 16 *KRT16*, Keratin, type II cytoskeletal 6B *KRT6B*, Annexin A1 *ANXA1*) and thus require stringent evaluation. Some of the unimodal genes and peptides were found to co-locate and showed a strong correlation when spatial omics modalities were combined through serial slices (such as Keratin, type I cytoskeletal 16 *KRT16*, Keratin, type II cytoskeletal 6B *KRT6B*, Annexin A1 *ANXA1*). These gene-

peptide candidates show the potential of multimodal biomarkers for hypoxia. While it is important to accurately detect hypoxia, multimodal biomarkers would also contribute to a more nuanced understanding of the underlying processes in hypoxic tumors than unimodal markers.

Although several unimodal and multimodal biomarker candidates were detected, several limitations need to be considered. First, whether the found markers are consistently verifiable in hypoxic regions need to be evaluated by additional data. As part of future work, it is planned to investigate the biomarkers on consecutive tissue slices using immunohistochemistry stains. Second, so far, the convolutional autoencoders were trained on a relatively small set of individual HNSCC tumor models. As a consequence, they may not generalize well to unknown tumor models. Therefore, it is import to train the models on a larger and more diverse set of samples. Third, the hypoxia annotations were derived from consecutive tissue slices. Inevitably, this will produce some biologically and technically inaccuracies. The former category includes inaccuracies due to the presence of different cells in consecutive slices. Among the latter, inaccuracies may be introduced by means of co-registration errors. While a more elaborate co-registration method may reduce inaccuracies further, the use of consecutive slices will likely remain a source of error. One a similar note, the fluorescence images were only partially annotated. As a consequence, some stains might appear to represent hypoxia when they are actually just artifacts. This issue may be resolved by increasing the number of (at least partially) annotated images to diminish sensitivity to false positive hypoxia labels. Forth, MSI and SPT data had to be collected on consecutive tissue slices. Therefore, a combination of MSI and SPT data results in comparable constraints as described in point three. Another methodological constraint includes that potential multimodal biomarkers are restricted to co-located genes and peptides. Genes and peptides with a low spatial correlation cannot be identified with the proposed integration strategy.

Considering the promising though preliminary results for the combination of spatial omics data, further steps are warranted. Currently, the integration strategy based on serial slices is prone to two errors. The first one can be attributed to the high-dimensional feature space which will likely produce statistical errors when comparisons are carried out. These errors might be overcome by the integration of latent features from the proposed convolutional autoencoders, instead of fusing the entire feature sets. The second error

is introduced due to the usage of consecutive tissue slices. To mitigate the effects of biological and technical errors, it was investigated whether convolutional neural networks can learn spatial omics data through H&E stains. If so, one spatial omics data may be learned and subsequently predicted on other H&E stains, e.g., from other spatial omics modalities. Exemplified on peptide information from MSI, this strategy yielded promising results when comparing actual and predicted peptide intensity values, especially when dimensionality-reduced data from autoencoders had been learned. Given the high amount of MSI data that is available for HNSCC tumor models in our group, the application of convolutional autoencoders for combining spatial omics data seems more promising than the consecutive slice approach. Furthermore, this approach is easily expandable to other spatial omics modalities. Also, a model trained on H&E stains for MSI prediction might be used for additional applications: Taking into account that H&E stains a routinely acquired in clinical practice, it would also facilitate the implementation of molecular biomarker screening even if only in an unimodal setup. Once multimodal omics data is available on the same tissue slice, the actual fusion of genetic information might be established through more elaborate methods than the correlation analysis proposed. One alternative could be incorporating contrastive learning. This will likely lead to an increased model complexity, with one major challenge remaining: enabling explainable results.

In summary, this work illustrated how the high-dimensional and highly correlated feature space of spatial omics data can be analyzed and utilized for biomarker discovery. It demonstrated that commonly derived feature importance measures of random forest models are not reliable in light of highly correlated features. In contrast, autoencoders extracted features while still enabling to track back the contributions of individual features, even though these models are often criticized as black boxes. Considering the ever-increasing size of (spatial) omics data as a result of higher resolutions, the usage of feature extraction methods like autoencoders will become evermore important.

7 Supplementary Material

7.1 Co-Registration and Patch Implementation

Auxiliary classes *MovingImage* and *ImageRep* are utilized for co-registration of FI or H&E images to spatial omics data.

Code Snippet 7.1: Excerpt of auxiliary class MovingImage

```
import numpy as np
import pandas as pd

import itk
from skimage import exposure

class MovingImage(DataProcessing):
    def __init__(self, params: ConfigParams):
        self.params = params
        self._moving = None # ImageRep
        self._coreg = None # ImageRep
        self.original = None


    def read(self, path, with_clahe = False):
        if not with_clahe:
            self.original = itk.imread(path, itk.F)
        else:
            pixel_type = itk.RGBPixel[itk.UC]
            original = itk.imread(path, pixel_type)
            arr = itk.GetArrayViewFromImage(original)
            gray_arr = cv2.cvtColor(arr, cv2.COLOR_BGR2GRAY)
            clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
            self.original = clahe.apply(gray_arr)

        return self

    def set_original(self, im):
        self.original = im
        return self


    def read_channel(self, path, channel):
        im = cv2.imread(path)[...,channel]
        self.original = np.asarray(im).astype(np.float32)

        return self

```

```
39      @property
40      def moving(self):
41          if self._moving is None:
42              raise Exception('Data is not build yet.')
43          return self._moving
44
45      @property
46      def coreg(self):
47          if self._coreg is None:
48              raise Exception('Data is not build yet.')
49          return self._coreg
50
51
52      def co_register(self, fixed, iterations = 2000, deform = False):
53          coreg, transform_map = self.moving.co_register(fixed, iterations, deform)
54          self._coreg = coreg
55          return transform_map
56
57      def match_histograms(self, img_rep):
58          self._moving = ImageRep(exposure.match_histograms(self.moving.rep, img_rep), self.moving.default_pixel)
59          return self
60
61
62      def apply_transform(self, transform_map):
63          self._coreg = self.moving.apply_transform(transform_map)
64          return self
65
66      def setup_moving(self, default_pixel, reshape = False):
67          img_norm = DataProcessing.normalize_min_max(self.original, 1)
68          self._moving = ImageRep(img_norm, default_pixel).transform(self.params, reshape)
69
70          self.original = None
71          return self
72
73
74      def crop(self, threshold: float = 0.001):
75          moving, coords = self.moving.crop(threshold)
76
77          self._moving = moving
78          self._coords = coords
79          return self
80
81      def apply_crop(self, dims):
82          self._moving = self.moving.apply_crop(dims)
83          return self
84
85      def rotate(self, angle, reshape):
86          self._moving = self.moving.rotate(angle, reshape)
87          return self
88
89      # methods applied on .coreg image, maybe rename for consistency reasons
90      def downsize(self, factor):
91          self._coreg = self._coreg.resize_by_factor(factor)
92          return self
93
94      def pad(self, padding):
95          self._coreg = self._coreg.pad(padding)
96          return self
97
98      def resize(self, interpolation = cv2.INTER_CUBIC):
99          factor = self.params.img_size / np.max(self.moving.rep.shape)
100         self._moving = self.moving.resize_by_factor(factor)
101         return self
```

Code Snippet 7.2: Excerpt of auxiliary classes ImageRep and ImageMask

```
1
2  from scipy import ndimage
3  import cv2
4  import itk
5
6  class ImageRep():
7      def __init__(self, rep, default_pixel):
8          self.rep = rep
9          self.default_pixel = default_pixel
10         self._mask = None
11
12     @property
13     def mask(self):
14         if self._mask is None:
15             self._mask = self.set_mask()
16         return self._mask
17
18     def set_mask_with_shape(self, shape, factor):
19         mask =  np.zeros(shape, dtype=np.float32)
20         spots = list(map(tuple, np.multiply(self.get_spots(), factor)))
21
22         max_coord_y = max(spots, key=lambda x: x[0])[0]
23         max_coord_x = max(spots, key=lambda x: x[1])[1]
24
25         if shape[0] < max_coord_y or shape[1] < max_coord_x:
26             return ImageMask(mask)
27         return ImageMask(mask).build(spots)
28
29     def set_mask(self):
30         mask = np.zeros(self.rep.shape, dtype=np.float32)
31         return ImageMask(mask).build(self.get_spots())
32
33     def get_spots(self):
34         if self.default_pixel == 1.0:
35             return list(map(tuple, np.argwhere(self.rep < .90)))
36         else:
37             return list(map(tuple, np.argwhere(self.rep > .10)))
38
39     def transform(self, params, reshape=True):
40         rep = self.rep
41         if params.rotate is not None:
42             rep = ndimage.rotate(rep, params.rotate, cval = self.default_pixel, reshape=reshape)
43         if params.flip is not None:
44             rep = cv2.flip(rep, params.flip)
45
46         return ImageRep(rep, self.default_pixel)
47
48     def get_sample_coords(self, threshold = 0.0001):
49         if self.default_pixel == 0.:
50             tissue_coords_x = np.where(np.greater(np.mean(self.rep, axis=0), self.default_pixel + threshold))[0]
51             tissue_coords_y = np.where(np.greater(np.mean(self.rep, axis=1), self.default_pixel + threshold))[0]
52         elif self.default_pixel == 1.:
53             tissue_coords_x = np.where(np.less(np.mean(self.rep, axis=0), self.default_pixel - threshold))[0]
54             tissue_coords_y = np.where(np.less(np.mean(self.rep, axis=1), self.default_pixel - threshold))[0]
55
56         return ImageDimensions(min(tissue_coords_x), max(tissue_coords_x), min(tissue_coords_y),  max(tissue_coords_y))
57
58     def cropped_max(self, threshold = 0.0001, border = 3):
59         coords = self.get_sample_coords(threshold)
60         return ImageRep(self.rep[:coords.y_max + border, :coords.x_max + border], self.default_pixel)
61
```

```
62      def crop(self, threshold = 0.0001):
63          coords = self.get_sample_coords(threshold)
64
65          return ImageRep(self.rep[coords.y_min:coords.y_max, coords.x_min:coords.x_max], self.default_pixel), coords
66
67      def apply_crop(self, dims):
68          return ImageRep(self.rep[dims.y_min:dims.y_max, dims.x_min:dims.x_max], self.default_pixel)
69
70      def apply_max_crop(self, yx):
71          return ImageRep(self.rep[:yx[0], :yx[1]], self.default_pixel)
72
73      def rotate(self, angle, reshape):
74          return ImageRep(ndimage.rotate(self.rep, np.math.degrees(angle), reshape=reshape, cval=self.default_pixel),
                self.default_pixel)
75
76      def resize_by_factor(self, scaling_factor):
77          return ImageRep(cv2.resize(self.rep, (0,0), fx=scaling_factor, fy=scaling_factor), self.default_pixel)
78
79      def resize_by_shape(self, shape, interpolation = cv2.INTER_AREA):
80          return ImageRep(cv2.resize(self.rep, shape, interpolation = interpolation), self.default_pixel)
81
82      def pad(self, padding):
83          padded = np.pad(self.rep, ((0, padding[0]), (0, padding[1])), mode='constant', constant_values=(self.default_pixel))
84          return ImageRep(padded, self.default_pixel)
85
86      def co_register(self, fixed, iterations = 2000, deform = False):
87          parameter_object = itk.ParameterObject.New()
88          parameterMap = parameter_object.GetDefaultParameterMap('affine')
89          parameterMap['DefaultPixelValue'] = [str(self.default_pixel)]
90          parameterMap['Registration'] = ['MultiResolutionRegistration'] # seems to be default
91          parameterMap['NumberOfResolutions'] = [str(4)] # seems to be default
92          parameterMap['Transform'] = ['SimilarityTransform']
93          parameterMap['AutomaticParameterEstimation'] = ['false']
94          parameterMap['AutomaticTransformInitialization'] = ['true']
95          parameterMap['AutomaticScalesEstimation'] = ['true']
96          parameterMap['MaximumNumberOfIterations'] = [str(iterations)]
97          parameter_object.AddParameterMap(parameterMap)
98
99          if deform: # experimental
100             bspline_parameter_map = parameter_object.GetDefaultParameterMap('bspline', 2)
101             bspline_parameter_map['DefaultPixelValue'] = [str(self.default_pixel)]
102             bspline_parameter_map['MaximumNumberOfIterations'] = [str(iterations/2)]
103             bspline_parameter_map['FinalBSplineInterpolationOrder'] = ['2']
104             bspline_parameter_map["FinalGridSpacingInPhysicalUnits"] = ['1']
105             bspline_parameter_map["NumberOfSamplesForExactGradient"] = ['8192']
106             parameter_object.AddParameterMap(bspline_parameter_map)
107
108         result_image, params = itk.elastix_registration_method(
109             fixed, self.rep,
110             output_directory = DIRECTORY,
111             log_file_name = "elastix_coreg", log_to_file = True,
112             parameter_object = parameter_object, log_to_console = False)
113
114         return ImageRep(result_image, self.default_pixel), params
115
116     def apply_transform(self, transform_params):
117         result_image = itk.transformix_filter(
118             self.rep,
119             transform_params,
120             log_to_console=False)
121
122         return ImageRep(result_image, self.default_pixel)
123
```

```
class ImageMask():
    def __init__(self, img):
        self.img = img
        self.default_pixel = 0.0

    def build(self, spots):
        int_coords = [(int(y), int(x)) for y, x in spots]
        for y, x in int_coords:
            self.img[y, x] = 1.0
        return self

    def get(self):
        return self.img

    def close_spots(self, kernel_size = (3,3), iterations = 3):
        kernel = np.ones(kernel_size, np.uint8)
        self.img = cv2.morphologyEx(self.img, cv2.MORPH_CLOSE, kernel, iterations=iterations)
        return self

    def dilate_spots(self, kernel_size = (2,2), iterations = 3):
        kernel = np.ones(kernel_size, np.uint8)
        self.img = cv2.morphologyEx(self.img, cv2.MORPH_DILATE, kernel, iterations=iterations)
        return self

    def calculate_dice_score(self, mask):
        intersection = np.sum(mask.img[self.img==1]) * 2.0
        dice = intersection / (np.sum(mask.img) + np.sum(self.img))
        return dice

class ImageDimensions:
    def __init__(self, x_min, x_max, y_min, y_max):
        self.x_min = x_min
        self.x_max = x_max
        self.y_min = y_min
        self.y_max = y_max

    def get_dims(self):
        return (self.y_max - self.y_min, self.x_max - self.x_min)
```

Classes *MSIDataset* and *SPTDataset* convert an annData [125] structure to images of dimensions (padded image size × padded image size × number of features). These images can afterwards further be split into patches using the class *Patches*, with the possibility to create overlapping patches.

Code Snippet 7.3: Auxiliary classes for creation of patches for some given spatial omics modality

```
import numpy as np
import pandas as pd
import anndata as ad

import tensorflow as tf
import matplotlib.pyplot as plt

class MSIDataset(SpatialDataset):

    def __init__(self, patch_size: int, n_features:int,
                 im_label: dict = {'HE': 0, 'FI': 1}, obs_label: dict = {"FI" : "fi", "HE": "he"}, obs_xy: list =
                     ["xLocation", "yLocation"], background:int = 0):
        super().__init__(patch_size, n_features, im_label, obs_label, obs_xy, background)

```

```
14  class SPTDataset(SpatialDataset):
15      def __init__(self, patch_size: int, n_features:int,
16                   im_label: dict = {'FI': 0}, obs_label: dict = {"FI" : "hypoxia"}, obs_xy: list = ["x", "y"], background:int
                         = 0):
17          super().__init__(patch_size, n_features, im_label, obs_label, obs_xy, background)
18
19  class SpatialDataset():
20      def __init__(self, patch_size: int, n_features:int,
21                   im_label: dict, obs_label: dict, obs_xy: list, background:int = 0):
22          self._images = SpatialImages(n_features, patch_size, im_label, obs_label, obs_xy, background)
23          self._patches = Patches(patch_size, n_features)
24          self._yatches = Patches(patch_size, len(im_label))
25          self._train = None
26          self._test = None
27
28      @property
29      def train(self):
30          if self._train is None:
31              raise Exception('Dataset is not build yet.')
32          return self._train
33
34      @property
35      def test(self):
36          if self._test is None:
37              raise Exception('Dataset is not build yet.')
38          return self._test
39
40      def build(self, train, test, create_patches = True):
41          train_images, train_labels = self._images.unfold_all_images(train)
42          self._train = SpatialData(self._patches, train_images, self._yatches, train_labels, create_patches)
43
44          if test is not None:
45              test_images, test_labels = self._images.unfold_all_images(test)
46              self._test = SpatialData(self._patches, test_images, self._yatches, test_labels)
47          return self
48
49      def overlapping_patches(self, stride:int, on_test: bool = False):
50          self._train = self._train.create_overlapping_patches(stride)
51          if on_test:
52              self._test = self._test.create_overlapping_patches(stride)
53          return self
54
55  class SpatialImages():
56      def __init__(self, n_features: int, padding_base: int, im_label: dict, obs_label: dict, obs_xy: list,
57                   background: int):
58          self.n_features = n_features
59          self.padding_base = padding_base
60          self.im_label = im_label
61          self.obs_label = obs_label
62          self.obs_xy = obs_xy
63          self.background = background
64
65      def unfold_all_images(self, adata : ad.AnnData) -> pd.Series :
66          batches = adata.obs.batch.value_counts()
67          images = []
68          names = []
69          labels = []
70          for name, value in batches.items():
71              batch = adata[adata.obs.batch == name]
72              im_range, xy_min = self.set_image_range(batch)
73              im, l = self.unfold_image(batch, im_range, xy_min)
74              images.append(im)
75              labels.append(l)
76              names.append(name)
77
78          return pd.Series(images, index = names), pd.Series(labels, index = names)
79
```

```
80      def set_image_range(self, batch : ad.AnnData):
81          sample = batch.obs[self.obs_xy]
82          xy_max = sample.max()
83          xy_min = sample.min() # need to be passed, default 0
84          im_range = xy_max - xy_min + 1
85
86          padded_size = self.padding_base * np.ceil(im_range.max()/self.padding_base).astype(int)
87          im_range = (padded_size, padded_size) # extract to other method and pass
88
89          return im_range, xy_min
90
91      def unfold_image(self, batch: ad.AnnData, im_range: tuple, xy_min: tuple):
92          sample = batch.obs[self.obs_xy]
93          data = batch.to_df()
94          im = np.full((*im_range, self.n_features), self.background, dtype=float)
95          l = np.full((*im_range, len(self.im_label)), 0, dtype = float)
96
97          for i in range(sample.shape[0]):
98              xy = sample.iloc[i] - xy_min
99
100             if not np.all(np.isnan(data.iloc[i])):
101                 im[tuple(xy)[::-1]] = data.iloc[i]
102
103             for j in self.im_label:
104                 l[tuple(xy)[::-1]][self.im_label[j]] = batch.obs[self.obs_label[j]].iloc[i]
105         return im, l
106
107 class Patches():
108     def __init__(self, patch_size : int, n_features : int):
109         self.patch_size = patch_size
110         self.no_patches = {}
111         self.n_features = n_features
112
113     def create_all(self, images : pd.Series):
114         images_patches = []
115         for name, im in images.items():
116             p, no_p = self.create(im, self.patch_size)
117             images_patches.append((name, p))
118             self.no_patches[name] = no_p
119
120         return pd.Series(dict(images_patches))
121
122     def create_all_with_stride(self, images : pd.Series, stride: int):
123         images_patches = []
124         for name, im in images.items():
125             p, no_p = self.create(im, stride)
126             images_patches.append((name, p))
127
128         return pd.Series(dict(images_patches))
129
130     def create(self, image: np.ndarray, stride: int):
131         patch_tensor = [1, self.patch_size, self.patch_size, 1] # without n_features
132         padded_size = image.shape[0]
133         no_patches = int(padded_size * padded_size / (self.patch_size * self.patch_size))
134         strides =  [1, stride, stride, 1]
135         patches = tf.image.extract_patches(images = tf.expand_dims(image, 0),
136                         sizes = patch_tensor,
137                         strides = strides,
138                         rates = [1, 1, 1, 1], padding='VALID')
139
140         patches = tf.reshape(patches, (patches.shape[1]*patches.shape[2], self.patch_size, self.patch_size, image.shape[2]))
141         return patches, patches.shape[0]
142
```

```
143     def plot_full_image(self, image, name, mz_value, path_prefix = "", inverse = False):
144         img = 1. - image[...,mz_value] if inverse else image[...,mz_value]
145         plt.figure(figsize=(4, 4))
146 
147         plt.imshow(img, vmin=0, vmax=1)
148         plt.title(name)
149         plt.colorbar()
150 
151         if path_prefix != "": plt.savefig(path_prefix + "_" + str(mz_value) + ".png", dpi = 300, facecolor = "white")
152         else: plt.show()
153 
154     def shape_with_no_patches_and_mz(self, patch, name, patch_size, n_features):
155         patch = np.reshape(patch, (self.no_patches[name], patch_size, patch_size, n_features))
156         n = int(np.sqrt(patch.shape[0]))
157         return patch, n
158 
159     def show_patch(self, image, patch, mz_value = 1):
160         plt.figure(figsize=(4, 4))
161         plt.imshow(image[patch,...,mz_value], vmin=0, vmax=1)
162         plt.colorbar()
163 
164     def show_patch_in_image(self, image, max_p = -1,  mz_value = 1, path_prefix = "", plot_show = True, inverse = False):
165         if max_p == -1: max_p = image.shape[0] - 1
166         n = int(np.sqrt(image.shape[0]))
167 
168         for i in range(0, max_p + 1):
169             ax = plt.subplot(n, n, i + 1)
170             img = 1. - image[i,...,mz_value] if inverse else image[i,...,mz_value]
171             plt.imshow(img, vmin=0, vmax=1)
172             plt.axis("off")
173         if path_prefix != "": plt.savefig(path_prefix + "_" + str(mz_value) + "_patched_" + str(max_p) + ".png", dpi = 300,
                facecolor = "white")
174         if plot_show: plt.show()
175 
176     def show_patched_image(self, patches : np.ndarray, name : str, mz_value = 1, max_p = -1, path_prefix = ""):
177         patch = patches[name]
178         (_, image) = self.get_image_from_patch(patch, name)
179         self.plot_full_image(image, name, mz_value, path_prefix)
180         self.show_patch_in_image(patch, mz_value, max_p, path_prefix)
181 
182     def get_image_from_patch(self, patch : pd.Series, name : str):
183         patch, n = self.shape_with_no_patches_and_mz(patch, name, patch.shape[2], patch.shape[3])
184         rows = np.split(patch, n, axis=0) # n x (n, patch_size, patch_size, n_features)
185         rows = [np.concatenate(np.moveaxis(x, 0, 0), axis = 1) for x in rows] # n x (patch_size, patch_size * n, n_features)
186         image = np.concatenate(rows, axis = 0) # (patch_size * n, patch_size * n, n_features)
187         return (name, image)
188 
189     def get_labeled_images_from_patches(self, blueprint : pd.Series, patches : np.ndarray):
190         index_split = np.cumsum([p.shape[0] for p in blueprint])
191         # caution, assuming patches are symmetric
192         value_split = np.split(patches, index_split[:-1])
193         patches = pd.Series(value_split, index = blueprint.index)
194         return patches, self.get_images_from_patches(patches)
195 
196     def get_images_from_patches(self, patches : pd.Series):
197         rebuild_images = [self.get_image_from_patch(p, name) for name, p in patches.items()]
198         return pd.Series(dict(rebuild_images))
199 
```

```
200 class SpatialData():
201     def __init__(self, patches: Patches, images:list, yatches:Patches, labels:list, create_patches = True):
202         self._patches = patches
203         self._yatches = yatches
204         if create_patches:
205             self.x = SpatialDataRep(images, self._patches.create_all(images))
206             self.y = SpatialDataRep(labels, self._yatches.create_all(labels))
207         else:
208             self.x = SpatialDataRep(images, None)
209             self.y = SpatialDataRep(labels, None)
210         self.z = None
211         self.dec = None
212 
213     def get_non_empty_patches(self):
214         non_zero_p = self.x.identify_non_zero_patches()
215         x, y = self.x.flatten(), self.y.flatten()
216         return x[non_zero_p], y[non_zero_p]
217 
218     def create_overlapping_patches(self, stride:int):
219         if stride != 0:  # != patch_size
220             # should be smaller than patch size
221             self.x._patches_overlap = self._patches.create_all_with_stride(self.x.images, stride)
222             self.y._patches_overlap = self._patches.create_all_with_stride(self.y.images, stride)
223         else:
224             self.x._patches_overlap = self.x.patches
225             self.y._patches_overlap = self.y.patches
226         return self
227 
228     def set_z_and_dec(self, args):
229         z, dec_patches = args
230         dec_patches, dec_images = self._patches.get_labeled_images_from_patches(self.x.patches, dec_patches)
231         self.z = z
232         self.dec = SpatialDataRep(dec_images, dec_patches)
233 
234 class SpatialDataRep():
235     def __init__(self, images, patches, patches_overlap = None):
236         self.images: SpatialImages = images
237         self.patches = patches
238         self._patches_overlap = patches_overlap
239 
240     @property
241     def patches_overlap(self):
242         if self._patches_overlap is None:
243             raise Exception('Patches have not build yet.')
244         return self._patches_overlap
245 
246     def flatten(self):
247         patches = self.patches if self._patches_overlap is None else self.patches_overlap
248         return np.concatenate(patches)
249 
250     def identify_non_zero_patches(self):
251         patches = self.flatten()
252         mean_value = np.mean(patches, axis = (1,2,3))
253         return np.where(mean_value != 0.)[0]
```

7.2 Convolutional Autoencoder Implementation

As initially, I experimented with various numbers of layers, the implementation of class *ConvAE* was designed such that the number of convolutional layers can be defined in a flexible way. The autoencoder takes a list of convolutional layers in the form of *n_channels* × *kernel_size* × *strides* for both, the encoder and decoder, to then build a composite of convolution, ReLU and batch normalization. An helper function was created to build a list of convolutional layers (named *AdjConvParams.build_power2_conv_layers_1x1_middle()*), by defining the channel number in the first and last hidden layer as power of two values and a step size. For example, conv_filter_max $= 1024$, conv_filter_min $= 16$, conv_filter_step $= 3$ would set up three convolutional layers of 1024 × kernel_size × strides, 128 × kernel_size × strides and 16 × kernel_size × strides.

Code Snippet 7.4: Convolutional autoencoder implementation

```
1  import tensorflow as tf
2  from tensorflow.keras import layers
3  import pandas as pd
4  import numpy as np
5  from abc import abstractmethod
6
7  class AE(object):
8      def __init__ (self, n_features : int):
9          self.n_features = n_features
10         self._encoder = None
11         self._decoder = None
12
13     @property
14     def encoder(self):
15         if self._encoder is None:
16             raise Exception('Encoder is not build yet.')
17         return self._encoder
18
19     @property
20     def decoder(self):
21         if self._decoder is None:
22             raise Exception('Decoder is not build yet.')
23         return self._decoder
24
25     @abstractmethod
26     def build(self, plot_vae = False):
27         pass
28
29     def predict(self, samples, verbose = 0):
30         p_flatten = np.concatenate(samples)
31         encoded = self._encoder.predict(p_flatten, verbose=verbose)
32         latent_z = np.squeeze(encoded)
33         decoded_data = self._decoder.predict(encoded, verbose=verbose)
34         return latent_z, decoded_data
35
```

```
36  class ConvAE(AE):
37      def __init__(self, n_features : int, patch_size : int, enc_conv_params : list, dec_conv_params : list, activation :
            str):
38          super().__init__(n_features)
39          self.patch_size = patch_size
40          self.enc_conv_params = enc_conv_params
41          self.dec_conv_params = dec_conv_params
42          self.activation = activation
43
44      def build(self, plot_vae = False):
45          inputs = tf.keras.Input(shape = (self.patch_size, self.patch_size, self.n_features))
46          h = inputs
47          for params in self.enc_conv_params:
48              h = self.build_conv2d_layers(params, h)
49          encoder = tf.keras.Model(inputs, h, name = 'encoder')
50          hdec = h
51          for params in self.dec_conv_params:
52              hdec = self.build_conv2dtranspose_layers(params, hdec)
53          decoder_outputs = layers.Conv2DTranspose(self.n_features,
54              (self.enc_conv_params[0].kernel, self.enc_conv_params[0].kernel),
55              strides=self.enc_conv_params[0].strides, activation=self.activation, padding="valid")(hdec)
56          decoder = tf.keras.Model(h, decoder_outputs, name = 'decoder')
57          self._encoder = encoder
58          self._decoder = decoder
59          return self
60
61      def build_conv2d_layers(self, param, inputs):
62          h = layers.Conv2D(param.filters, (param.kernel, param.kernel), strides=param.strides, padding="valid")(inputs)
63          h = layers.BatchNormalization()(h)
64          h = layers.ReLU()(h)
65          return(h)
66
67      def build_conv2dtranspose_layers(self, param, inputs):
68          hdec = layers.Conv2DTranspose(param.filters, (param.kernel, param.kernel), strides=param.strides,
                padding="valid")(inputs)
69          hdec = layers.BatchNormalization()(hdec)
70          hdec = layers.ReLU()(hdec)
71          return(hdec)
72
73  # msi_extension.py
74  class AdjConvParams(ae_vae.ConvParams):
75      @staticmethod
76      def build_power2_conv_layers_1x1_middle(conv_filter_max, conv_filter_min, conv_filter_step, kernel, stride,
            indices_for_x1):
77          enc_filters = []
78          dec_filters = []
79          dim = conv_filter_max
80          i = 0
81          step = 0
82          while dim >= conv_filter_min:
83              if step % conv_filter_step == 0:
84                  if i in indices_for_x1:
85                      enc_filters.append(ae_vae.ConvParams(dim, 1, 1))
86                      if i + 1 in indices_for_x1:
87                          dec_filters.insert(0, ae_vae.ConvParams(dim, 1, 1))
88                      else:
89                          dec_filters.insert(0, ae_vae.ConvParams(dim, kernel, stride))
90                  else:
91                      enc_filters.append(ae_vae.ConvParams(dim, kernel, stride))
92                      # depends if we start with 2x1 or with 1x1
93                      # start with 2x1
94                      dec_filters.insert(0, ae_vae.ConvParams(dim, 1, 1))
95
96                  i = i + 1
97              step = step + 1
98              dim = int(dim / 2)
99          return enc_filters, dec_filters[1:]
100
```

```
101 class ConvParams():
102     def __init__(self, filters, kernel, strides):
103         self.filters = filters
104         self.kernel = kernel
105         self.strides = strides
```

In the proposed recovery method, *data* reflects the input patches, *dummy_tensor* denotes a zero- or ones-filled tensor with the shape of *data*, and *latent_feature_id* represents the latent feature of interest. It is expected that patch representations of all samples are stored as TFRecordDataset. To define which patches belong to which sample, a list of indices, named *index_split*, needs to passed.

Code Snippet 7.5: Recovery method

```
1  import argparse
2  import os
3  os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
4  import tensorflow as tf
5  tf.config.threading.set_intra_op_parallelism_threads(1)
6  tf.config.threading.set_inter_op_parallelism_threads(1)
7  gpus = tf.config.experimental.list_physical_devices('GPU')
8  if gpus:
9      for gpu in gpus:
10         tf.config.experimental.set_memory_growth(gpu, True)
11
12 import numpy as np
13 import pandas as pd
14 import scipy.stats
15 import h5py
16 import skimage.transform as st
17 import gc
18 # packages created by Verena Bitto
19 from automsi import ae_utils
20 from autospt import msi_extension, spt_datasets
21
22 def parse_args():
23     parser = argparse.ArgumentParser()
24     parser.add_argument('--tfrec_path', type=str, help='Path to tfrecord files')
25     parser.add_argument('--ae_export', type=str, help='Path to experiment')
26     parser.add_argument('--experiment', type=str, help='Name to experiments from which autoencoder parameters can be
           derived.')
27     parser.add_argument('--scaler', type=str, help='Scaler to normalize spatial data.')
28     parser.add_argument('--samples', type=ae_utils.split_at_semicolon, help='Sample names to encode with the autoencoder,
           separated by ;')
29     # don't change order
30     parser.add_argument('--backfill', type=str, default="ones", help='Noise values to fill features.')
31     parser.add_argument('--latent_feature_id', type=int, help='Id of latent feature to recover information.')
32     parser.add_argument('--n_patches', type=int, help='Number of patches to process.')
33     parser.add_argument('--index_split', type=ae_utils.split_int_at_semicolon, help='Denotes which patches belong to a
           sample.')
34     parser.add_argument('--spatial_data', type=str, default="msi", help='Define spatial omics data (msi or spt).')
35     return parser.parse_args()
36
```

```
37  def update_tensor(feature_patches, dummy_tensor, feature_id):
38      transposed_tensor = tf.transpose(feature_patches, perm=[1, 2, 3, 0])
39
40      if feature_id == 0:
41          updated_tensor = tf.concat([transposed_tensor, dummy_tensor[..., feature_id+1:]], axis=-1)
42      elif feature_id == n_features-1:
43          updated_tensor = tf.concat([dummy_tensor[..., :feature_id], transposed_tensor], axis=-1)
44      else:
45          updated_tensor = tf.concat([dummy_tensor[..., :feature_id], transposed_tensor, dummy_tensor[..., feature_id+1:]],
                axis=-1)
46
47      return updated_tensor
48
49  def rebuild_image(sample, feature_id):
50      patch = np.reshape(sample[...,feature_id], (sample.shape[0], sample.shape[1], sample.shape[2], 1))
51      n = int(np.sqrt(patch.shape[0]))
52      rows = np.split(patch, n, axis=0) # n x (n, patch_size, patch_size, n_features)
53      rows = [np.concatenate(np.moveaxis(x, 0, 0), axis = 1) for x in rows] # n x (patch_size, patch_size * n, n_features)
54      image = np.concatenate(rows, axis = 0) # (patch_size * n, patch_size * n, n_features)
55      return image
56
57  def rebuild_images(patches, feature_id):
58      series = np.split(patches, args.index_split[:-1])
59      images = [rebuild_image(sample, feature_id) for sample in series]
60      return images
61
62  def compute_correlation_coefficient(latent_im, org_im):
63      org_im = st.resize(org_im, latent_im.shape)
64
65      # ignore background pixels for comparison
66      org_im = np.where(org_im==0.0, np.nan, org_im)
67      im_nans = np.isnan(org_im)
68
69      cor = 0.0
70      cm = scipy.stats.spearmanr(latent_im[~im_nans].flat, org_im[~im_nans].flat)
71      if cm.pvalue < 0.05:
72          cor = cm.correlation
73      return cor
74
```

```
75  def main(args):
76      cor = np.zeros(n_features)
77
78      for feature_id, batch in enumerate(data):
79          # updated_tensor only contains information at position [...,feature_id]
80          updated_tensor = update_tensor(batch, dummy_tensor, feature_id)
81          # encode modified patches
82          z = exp.conv_ae.encoder.predict(updated_tensor, verbose = 2)
83          _ = gc.collect()
84          # rebuild image from latent feature of interest
85          z_images = rebuild_images(z, args.latent_feature_id)
86          # and compare it against its original
87          org_images = rebuild_images(updated_tensor, feature_id)
88
89          sample_cor = np.zeros(len(args.index_split))
90          np.seterr(divide='ignore', invalid='ignore')
91
92          samples = len(args.index_split)
93          for s in range(samples):
94              sample_cor[s] = compute_correlation_coefficient(z_images[s], org_images[s])
95
96          mean_sample_cor = 0. if np.any(sample_cor == 0.) else np.mean(sample_cor)
97          cor[feature_id] = mean_sample_cor
98
99
100     file = ["cor_info_complete"] + list(vars(args).values())[5:6+1]
101     h5f = h5py.File(os.path.join(args.ae_export, args.experiment, '_'.join(str(n) for n in file) + "_0_" + n_features +
            ".h5") , 'w')
102     h5f.create_dataset("cor", data=cor)
103     h5f.close()
104
105 if __name__ == '__main__':
106     args = parse_args()
107     #exp = ae_vae.ConvAEExperiment(args.experiment)
108     exp = msi_extension.SPTExperiment(args.experiment)
109     n_features = 10913
110     exp.build(n_features, args.ae_export, args.experiment)
111
112     BATCH_SIZE = 1
113     tfr_files_data =  [args.tfrec_path + args.samples + ".tfrecords"]
114
115     if args.backfill == "zeros":
116         dummy_tensor = tf.zeros(shape=(args.n_patches, exp.patch_size, exp.patch_size, n_features), dtype=tf.float32)
117     else:
118         dummy_tensor = tf.ones(shape=(args.n_patches, exp.patch_size, exp.patch_size, n_features), dtype=tf.float32)
119
120     data = (tf.data.TFRecordDataset(tfr_files_data)
121               .map(spt_datasets.parse_tfr_x_by_feature, num_parallel_calls=1)
122               .batch(BATCH_SIZE)
123               .prefetch(buffer_size=64)
124              )
125
126     main(args)
```

The trained convolutional autoencoder models are stored by a unique identifier, such that the name can be used to parse the (hyper)parameters of a model for loading the trained weights. This is achieved through the classes *ConvAEExperiment* and *SPTExperiment*. The latter class is used for spatial transcriptomics experiments, where also the position of the $1x1$ kernels was encoded. Class *Experiment* is used for non-dimensionality reduced data.

Code Snippet 7.6: Auxiliary classes for loading trained models

```
import numpy as np
from sklearn import preprocessing
import pandas as pd
import anndata as ad
import joblib
# ae_vae.py
class Experiment():
    def __init__(self, experiment, scaler = None, suffix = ""):
        params = experiment.split("_")
        self.file = "_".join(params[0:2])
        self.patch_size = int(params[2])
        self.scaler = scaler
        self.suffix = suffix

    def build(self, n_features, path, experiment_with_suffix):
        return self

    def split_x_y(self, patches):
        self.x, self.y = patches.x_mean[patches.poi], patches.y_mean[patches.poi]
        return self

    def set_data(self):
        self.z = self.x
        return self

    def set_reduced_data(self, features):
        mz_idx = np.random.choice(self.x.shape[1], features, replace=False)
        self.z = self.x[..., mz_idx]
        return self

    def transform(self, z):
        if self.scaler is None:
            return z
        return self.scaler.transform(z)

    def get_mean_data(self, patches):
        mean_data = np.mean(patches, axis = (1,2))
        if self.scaler is not None:
            mean_data = mean_data.reshape(-1, 1)
            self.scaler.partial_fit(mean_data)
        return mean_data

class ConvAEExperiment(Experiment):
    def __init__(self, experiment, scaler = None, suffix = ""):
        super().__init__(experiment, suffix, scaler)
        params = experiment.split("_")
        # patch_size = params[2]
        self.activation = params[3]
        self.kernel = int(params[5])
        self.stride = int(params[6])
        self.conv_filter_max = int(params[7])
        self.conv_filter_min = int(params[8])
        self.conv_filter_step = int(params[9])
        self.conv_filter_1x1_pos = 0
        self.samples_no = "_".join(params[11:14+1])
        self.mode = params[15]

        self.enc_conv_params, self.dec_conv_params = AdjConvParams.build_power2_conv_layers_1x1_middle(self.conv_filter_max,
            self.conv_filter_min, self.conv_filter_step, self.kernel, self.stride, [self.conv_filter_1x1_pos])

    def build(self, n_features, path, experiment_with_suffix):
        conv_ae = ConvAE(n_features, self.patch_size, self.enc_conv_params, self.dec_conv_params,
            self.activation).build(False)
        conv_ae.encoder.load_weights(path + "weights/" + experiment_with_suffix + '_encoder_model_weights.h5')
        conv_ae.decoder.load_weights(path + "weights/" + experiment_with_suffix + '_decoder_model_weights.h5')
        self.conv_ae = conv_ae
        return self
```

```
66
67      def build_z(self, patches):
68          z = self.conv_ae.encoder.predict(patches, verbose=0)
69          _ = gc.collect()
70          mean_z = np.mean(z, axis = (1,2))
71          return mean_z
72
73      def split_x_y(self, patches):
74          self.x, self.y = patches.x[patches.poi], patches.y_mean[patches.poi]
75          return self
76
77      def set_data(self):
78          self.z = self.build_z(self.x)
79          return self
80
81      def get_mean_data(self, patches):
82          mean_z = self.build_z(patches)
83          self.scaler.partial_fit(mean_z)
84          return mean_z
85
86  # msi_extension.py
87  class SPTExperiment(ae_vae.ConvAEExperiment):
88      def __init__(self, experiment, scaler = None, suffix = ""):
89          super().__init__(experiment, suffix, scaler)
90
91          params = experiment.split("_")
92          self.activation = params[4]
93          self.kernel = int(params[6])
94          self.stride = int(params[7])
95          self.conv_filter_max = int(params[8])
96          self.conv_filter_min = int(params[9])
97          self.conv_filter_step = int(params[10])
98          self.conv_filter_1x1_pos = int(params[11])
99          self.samples_no = "_".join(params[13:16+1])
100         self.mode = params[16]
101
102         self.enc_conv_params, self.dec_conv_params = AdjConvParams.build_power2_conv_layers_1x1_middle(self.conv_filter_max,
                self.conv_filter_min, self.conv_filter_step, self.kernel, self.stride, [self.conv_filter_1x1_pos])
103
104 # ae_processing.py
105 def normalize_with_obs(scale_fu, data: ad.AnnData, obs: pd.Series):
106     data_df = data.to_df()
107     norm = pd.DataFrame(scale_fu(data_df.values), columns=data_df.columns, index=data_df.index)
108     return ad.AnnData(norm, obs=obs)
109
110 def normalize_train_test_using_scaler(train : ad.AnnData, test : ad.AnnData, path_prefix : str, train_obs : pd.DataFrame,
      test_obs : pd.DataFrame) -> (ad.AnnData, ad.AnnData):
111     if train_obs.empty:
112         train_obs = train.obs.copy()
113     if test is not None and test_obs.empty:
114         test_obs = test.obs.copy()
115
116     scaler = joblib.load(path_prefix + "_scaler.gz")
117     train = normalize_with_obs(scaler.transform, train, train_obs)
118     if test is not None:
119         test = normalize_with_obs(scaler.transform, test, test_obs)
120     return train, test
121
122 # msi_extension.py
123 def read_spt_from_adata(path: str) -> (ad.AnnData, pd.Index, int):
124     adata = ad.read(path)
125     genes = adata.var_names
126     n_features = adata.n_vars
127     return adata, genes, n_features
```

In the method *reconstruction_error()* of class *CustomLoss*, x and x_pred denote the input data and predicted data, respectively, y holds the hypoxia labels and *threshold* defines the cutoff value for hypoxia.

Code Snippet 7.7: Training of convolutional autoencoder

```
class CustomLoss():

    @staticmethod
    def reconstruction_base(sdiff):
        return tf.reduce_mean(
            tf.reduce_mean(sdiff, axis = (1,2)),
        axis = 0) # reduce batch


    @staticmethod
    def post_norm(error, patch_size):
        patch_size = patch_size * patch_size
        error = error / tf.cast(patch_size, tf.float32)
        return error


    @staticmethod
    def reconstruction_error(x, x_pred, weight):
        sdiff = tf.math.abs(x - x_pred)
        sdiff = tf.multiply(sdiff, weight)

        reconstruction_err = CustomLoss.post_norm(CustomLoss.reconstruction_base(
           tf.reduce_sum(sdiff, axis=(3)) # reduce m/z values
        ), tf.shape(x)[1])

        return reconstruction_err


    @staticmethod
    def supervised_error(x, x_pred, y, threshold):
        # retains x.shape, e.g., (batch_size, 3, 3, 18735)
        sdiff = tf.math.abs(x - x_pred) # retains x.shape

        zero_mask =  tf.math.equal(tf.reduce_mean(x, axis=3), 0.0)
        non_zero_mask =  tf.logical_not(zero_mask)

        # retains y.shape, e.g., (batch_size, 3, 3)
        y_mask = tf.math.greater(y, threshold)
        # only consider non-empty pixel
        y_mask = tf.logical_and(non_zero_mask, y_mask)
        # down on pixel level, e.g., (matching_pixel, 18735)
        sdiff_weighted = tf.boolean_mask(sdiff, y_mask)

        is_shape_zero = tf.reduce_all(tf.equal(tf.shape(sdiff_weighted)[0], 0))

        def calculate_supervised_loss():
             return tf.reduce_sum(sdiff_weighted, axis=1)

        def return_empty_tensor():
            return tf.zeros(1, dtype=tf.float32)

        # down to single error per pixel, e.g., (error_per_pixel,)
        supervised_error = tf.cond(is_shape_zero, return_empty_tensor, calculate_supervised_loss)
        supervised_error = tf.reduce_mean(supervised_error)

        return supervised_error

```

```
58  class WeightedAETrainer(tf.keras.Model, CustomLoss):
59      def __init__(self, encoder, decoder, weight, **kwargs):
60          super(WeightedAETrainer, self).__init__(**kwargs)
61          self.encoder = encoder
62          self.decoder = decoder
63          self.weight = weight
64          self.reconstruction_loss_tracker = tf.keras.metrics.Mean(name="reconstruction_loss")
65
66      @property
67      def metrics(self):
68          return [
69              self.reconstruction_loss_tracker,
70          ]
71
72      def do_step(self, data):
73          if isinstance(data, tuple):
74              data = data[0]
75
76          x = data
77
78          # mask pixels which are zero
79          zero_mask = tf.math.equal(x, 0.0)
80          # inverse to get non-zero pixels
81          non_zero_mask =  tf.logical_not(zero_mask)
82          # set non-zero pixels to 1, and zero-pixel to 0
83          non_zero_mask = tf.cast(non_zero_mask, tf.float32)
84          # multiply by pixel weight, add one to essentially disable weighting in case of weight is set to 0
85          pixel_weight = tf.math.add(tf.math.multiply(non_zero_mask, self.weight), 1)
86
87          z = self.encoder(x)
88          x_pred = self.decoder(z)
89
90          reconstruction_loss = self.reconstruction_error(tf.cast(x, tf.float32), x_pred, pixel_weight)
91          return x_pred, reconstruction_loss
92
93      def train_step(self, data):
94          with tf.GradientTape() as tape:
95              _, reconstruction_loss = self.do_step(data)
96
97          grads = tape.gradient(reconstruction_loss, self.trainable_weights)
98          self.optimizer.apply_gradients(zip(grads, self.trainable_weights))
99
100         self.reconstruction_loss_tracker.update_state(reconstruction_loss)
101
102         return {
103             "reconstruction_loss": self.reconstruction_loss_tracker.result(),
104         }
105
106     def test_step(self, data):
107         _, reconstruction_loss = self.do_step(data)
108
109         self.reconstruction_loss_tracker.update_state(reconstruction_loss)
110         return {
111             "reconstruction_loss": reconstruction_loss,
112
113         }
114
```

```
115 class SemiSupervisedAETrainer(WeightedAETrainer):
116     def __init__(self, encoder, decoder, weight, supervised_threshold, **kwargs):
117         super().__init__(encoder, decoder, weight, **kwargs)
118         self.supervised_threshold = supervised_threshold
119         self.supervised_loss_tracker = tf.keras.metrics.Mean(name="supervised_loss")
120         self.total_loss_tracker = tf.keras.metrics.Mean(name="total_loss")
121 
122     @property
123     def metrics(self):
124         return [
125             self.supervised_loss_tracker,
126             self.total_loss_tracker,
127         ]
128 
129     def do_step(self, data):
130         x_pred, reconstruction_loss = super().do_step(data)
131 
132         x, y = data
133 
134         supervised_loss = self.supervised_error(tf.cast(x, tf.float32), x_pred, tf.cast(y, tf.float32),
                self.supervised_threshold)
135 
136         total_loss = reconstruction_loss + supervised_loss
137         return reconstruction_loss, supervised_loss, total_loss
138 
139     def train_step(self, data):
140         with tf.GradientTape() as tape:
141             reconstruction_loss, supervised_loss, total_loss = self.do_step(data)
142 
143         grads = tape.gradient(total_loss, self.trainable_weights)
144         self.optimizer.apply_gradients(zip(grads, self.trainable_weights))
145 
146         self.reconstruction_loss_tracker.update_state(reconstruction_loss)
147         self.supervised_loss_tracker.update_state(supervised_loss)
148         self.total_loss_tracker.update_state(total_loss)
149 
150         return {
151             "reconstruction_loss": self.reconstruction_loss_tracker.result(),
152             "supervised_loss": self.supervised_loss_tracker.result(),
153             "loss": self.total_loss_tracker.result(),
154         }
155 
156     def test_step(self, data):
157         reconstruction_loss, supervised_loss, total_loss = self.do_step(data)
158 
159         self.reconstruction_loss_tracker.update_state(reconstruction_loss)
160         self.supervised_loss_tracker.update_state(supervised_loss)
161         self.total_loss_tracker.update_state(total_loss)
162 
163         return {
164             "reconstruction_loss": reconstruction_loss,
165             "supervised_loss": supervised_loss,
166             "loss": total_loss,
167 
168         }
```

In Code Snippet 7.8, it is expected that the patches to train the convolutional autoencoder are stored as TFRecordDataset (see Code Snippet 7.9).

Code Snippet 7.8: Example of setup and training of convolutional autoencoder

```
import os
import argparse
import sys

import tensorflow as tf
tf.config.threading.set_intra_op_parallelism_threads(1)
tf.config.threading.set_inter_op_parallelism_threads(1)
gpus = tf.config.experimental.list_physical_devices('GPU')
# Enable memory growth for each GPU
if gpus:
    for gpu in gpus:
        tf.config.experimental.set_memory_growth(gpu, True)

from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt

# packages created by Verena Bitto
from automsi import ae_vae, ae_utils, ae_plots
from autospt import msi_extension, spt_datasets

BATCH_SIZE = 32

def parse_args():
    parser = argparse.ArgumentParser()
    parser.add_argument('--tfrec_path', type=str, help='Path to tfrecords files')
    parser.add_argument('--ae_export', type=str, help='Path where results directory will be created')
    parser.add_argument('--train_samples', type=ae_utils.split_at_semicolon, help='Sample names to train autoencoder,
        separated by ;')
    parser.add_argument('--test_samples', type=ae_utils.split_at_semicolon, help='Sample names to test autoencoder,
        separated by ;')
    parser.add_argument('--h5ad_files', type=str, default="cal336o1t0_15", help='File name, h5ad file without file suffix')

    # autoencoder params
    parser.add_argument('--patch_size', type=int, default=3, help='Patch size of convolutional layers')
    # not used when data is loaded as TFRecordDataset
    parser.add_argument('--overlapping_patches', type=int, default=1, help='Stride for creating overlapping patches')
    parser.add_argument('--activation', type=str, default="sigmoid", help='Last activation function')
    parser.add_argument('--weight', type=int, default=0, help='Allows weighting of non-background pixels')
    parser.add_argument('--kernel', type=int, default=2, help='Kernel size of convolutional layers')
    parser.add_argument('--stride', type=int, default=1, help='Stride of convolutional layers')
    parser.add_argument('--conv_filter_max', type=int, default=1024, help='Node size of first hidden layer')
    parser.add_argument('--conv_filter_min', type=int, default=64, help='Node size of last hidden layer')
    parser.add_argument('--conv_filter_step', type=int, default=4, help='Step size to define number of hidden layers between
        max and min, consider 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8')
    parser.add_argument('--conv_filter_1x1_pos', type=int, default=0, help='Position of 1x1 kernels')
    parser.add_argument('--epochs', type=int, default=25, help='Number of epochs to train autoencoder')

    parser.add_argument('--suffix', type=str, help='Suffix, e.g., to enumerate autoencoder runs (_001 to _010)')
    parser.add_argument('--mode', type=str, default="unsupervised", help='Mode of autoencoder training, i.e., unsupervised
        or semi-supervised')
    parser.add_argument('--semi_threshold', type=float, default=0.6, help='Cutoff value for labeled pixels, only used if
        mode = semi-supervised')
    return parser.parse_args()

def plot_history(history, key: str, max_loss_scale: int, max_kl_scale: int, plot_val: bool):
    ae_plots.plot_history(history, key, [0, max_loss_scale], plot_val, path_prefix + "/")

```

```
53  def main():
54      enc_conv_params, dec_conv_params = msi_extension.AdjConvParams.build_power2_conv_layers_1x1_middle(args.conv_filter_max,
55          args.conv_filter_min, args.conv_filter_step, args.kernel, args.stride, [args.conv_filter_1x1_pos])
56      conv_ae = ae_vae.ConvAE(n_features, args.patch_size, enc_conv_params, dec_conv_params, args.activation).build()
57
58      if args.mode == "semi-supervised":
59          learning_rate = 1e-4
60          ae = ae_vae.SemiSupervisedAETrainer(conv_ae.encoder, conv_ae.decoder, args.weight, args.semi_threshold)
61          ae.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate))
62
63          history = ae.fit(train,
64                           epochs = args.epochs, batch_size = BATCH_SIZE, shuffle = True, verbose=2,
65                           callbacks=(EarlyStopping(monitor="val_loss", patience=30, restore_best_weights=True)),
66                           validation_data = test,
67                       )
68          plot_history(history, "loss", 3000, 50, True)
69      elif args.mode == "unsupervised":
70          learning_rate = 1e-4
71          ae = ae_vae.WeightedAETrainer(conv_ae.encoder, conv_ae.decoder, args.weight)
72          ae.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate))
73
74          history = ae.fit(train,
75                           epochs = args.epochs, batch_size = BATCH_SIZE, shuffle = True,    verbose=2,
76                           callbacks=(EarlyStopping(monitor="val_reconstruction_loss", patience=args.epochs,
                                 restore_best_weights=True)),
77                           validation_data = (test, None),
78                          )
79
80          plot_history(history, "reconstruction_loss", 3000, 50, True)
81
82      if write_weights:
83          ae.encoder.save_weights(args.ae_export + "weights/" + prefix + '_encoder_model_weights.h5')
84          ae.decoder.save_weights(args.ae_export + "weights/" + prefix + '_decoder_model_weights.h5')
85
86  if __name__ == '__main__':
87      args = parse_args()
88      naming = list(vars(args).values())[5:17] + [args.mode] + [args.suffix]
89      n_features = 10913
90      prefix = '_'.join(str(n) for n in naming)
91      path_prefix = os.path.join(args.ae_export, prefix)
92      print("Creating directory for: " + prefix)
93      os.mkdir(path_prefix)
94
95      tfr_files_train =  [args.tfrec_path + args.train_samples + ".tfrecords"]
96      tfr_files_test =  [args.tfrec_path + args.test_samples + ".tfrecords"]
97
98      train = (tf.data.TFRecordDataset(tfr_files_train)
99               .map(spt_datasets.parse_complete_tfr, num_parallel_calls=1)
100              .batch(BATCH_SIZE)
101              .prefetch(buffer_size=4)
102             )
103
104     test = (tf.data.TFRecordDataset(tfr_files_train)
105             .map(spt_datasets.parse_complete_tfr, num_parallel_calls=1)
106             .batch(BATCH_SIZE)
107             .prefetch(buffer_size=4)
108            )
109
110     write_weights = True
111     main()
```

Code Snippet 7.9: Auxiliary classes to store and load TFRecordDatasets.

```
import numpy as np
import tensorflow as tf

# spt_datasets.py
def map_to_bytes_feature(value):
    return tf.train.Feature(bytes_list=tf.train.BytesList(value=[value.numpy()]))

def map_to_int64_feature(value):
    return tf.train.Feature(int64_list=tf.train.Int64List(value=[value]))

def map_to_float64_feature(value):
    return tf.train.Feature(int64_list=tf.train.FloatList(value=[value]))

def to_complete_tensors(x, y):
    data = {
        'x': map_to_bytes_feature(tf.io.serialize_tensor(x)),
        'y': map_to_bytes_feature(tf.io.serialize_tensor(y)),
        'patch_size': map_to_int64_feature(x.shape[1]),
        'n_features': map_to_int64_feature(x.shape[2])
    }
    return tf.train.Example(features=tf.train.Features(feature=data))

def to_tensors_x_by_feature(x):
    data = {
        'x': map_to_bytes_feature(tf.io.serialize_tensor(x)),
        'samples': map_to_int64_feature(x.shape[0]),
        'patch_size': map_to_int64_feature(x.shape[1])
    }
    return tf.train.Example(features=tf.train.Features(feature=data))

# note: cardinality is unkown
def parse_complete_tfr(serialized):
    data = {
      'x' : tf.io.FixedLenFeature([], tf.string),
      'y' : tf.io.FixedLenFeature([], tf.string),
      'patch_size' : tf.io.FixedLenFeature([], tf.int64),
      'n_features' : tf.io.FixedLenFeature([], tf.int64)
    }
    tfr = tf.io.parse_single_example(serialized, data)
    patch_size = tfr['patch_size']
    n_features = tfr['n_features']

    x = tf.io.parse_tensor(tfr['x'], out_type=tf.float32)
    x = tf.reshape(x, shape=[patch_size, patch_size, n_features])

    y = tf.io.parse_tensor(tfr['y'], out_type=tf.float32)
    y = tf.reshape(y, shape=[patch_size, patch_size])

    return x, y

def parse_tfr_x_by_feature(serialized):
    data = {
      'x' : tf.io.FixedLenFeature([], tf.string),
      'samples' : tf.io.FixedLenFeature([], tf.int64),
      'patch_size' : tf.io.FixedLenFeature([], tf.int64)
    }

    tfr = tf.io.parse_single_example(serialized, data)
    samples = tfr['samples']
    patch_size = tfr['patch_size']

    x = tf.io.parse_tensor(tfr['x'], out_type=tf.float32)
    x = tf.reshape(x, shape=[samples, patch_size, patch_size])
    return x

```

```
66  def write_tfr(x, y, path: str):
67      path = path + ".tfrecords"
68      with tf.io.TFRecordWriter(path) as file_writer:
69          for patch in range(x.shape[0]):
70              out = to_complete_tensors(x[patch,...], y[patch,...])
71              file_writer.write(out.SerializeToString())
72
73  def write_tfr_x_by_feature(x, path: str):
74      path = path + ".tfrecords"
75      with tf.io.TFRecordWriter(path) as file_writer:
76          for f in range(x.shape[3]):
77              out = to_tensors_x_by_feature(x[...,f])
78              file_writer.write(out.SerializeToString())
```

The example illustrates the processing of SPT data. Classes *Experiment* is used to train the random forest regression model on the raw omics data. Class *SPTExperiment* is utilized to encode patches by means of a previously trained convolutional autoencoder. For MSI, the code only differs in using the corresponding MSI classes when loading the data. Both, MSI and SPT data are typically built as sub classes of a common super class. For example, classes *SPTDataset* and *MSIDataset* inherit the variables and methods of class *SpatialDataset* (see Code Snippet 7.3).

Code Snippet 7.10: Example to perform regression on raw or dimensionality-reduced data

```
1  import argparse
2  import os
3  import tensorflow as tf
4  tf.config.threading.set_intra_op_parallelism_threads(1)
5  tf.config.threading.set_inter_op_parallelism_threads(1)
6  import numpy as np
7  import pandas as pd
8  import anndata as ad
9  import random
10 from sklearn.model_selection import cross_validate, ShuffleSplit
11
12 # packages implemented by Verena Bitto
13 from automsi import ae_preprocessing, ae_vae, ae_images, ae_utils, ae_rf
14 from autospt import *
15
16 N_TREES = 1000
17 MTRY = "sqrt"
18
19 def parse_args():
20     parser = argparse.ArgumentParser()
21     parser.add_argument('--h5ad_path', type=str, help='Path to h5ad files')
22     parser.add_argument('--ae_export', type=str, help='Path to experiment')
23     parser.add_argument('--experiment', type=str, help='Name to experiments from which autoencoder parameters can be
           derived.')
24     parser.add_argument('--scaler', type=str, help='Scaler to normalize MSI data.')
25     parser.add_argument('--samples', type=ae_utils.split_at_semicolon, help='Sample names to train the RF regressor,
           separated by ;')
26     parser.add_argument('--label', type=str, default="FI", help='Label used for training RF regressor.')
27     parser.add_argument('--overlapping_patches', type=int, default=1, help='Stride information for overlapping patches.')
28     parser.add_argument('--cutoff', type=float, help='Mean cutoff value to define a single patch as being hypoxic (or any
           other label), value between [0, 1].')
29     parser.add_argument('--other_fraction', type=float, help='Defines the fraction of non-hypoxic patches to be sampled,
           value between [0, 1].')
30     parser.add_argument('--mode', type=str, default="conv_ae", help='Whether to encode patches before training RF regressor
           or not (conv_ae or original).')
31     parser.add_argument('--features', type=int, default=18735, help='Number of features to use, for original mode only.')
32     parser.add_argument('--scoring', type=str, default="r2", help='Scoring metric for RF regressor.')
33     parser.add_argument('--suffix_from', type=str,  default="_001", help='Can be used to evaluate multiple experiments at
           once, expected to be of kind "_{0-9}3".')
34     parser.add_argument('--suffix_to', type=str,  default="_001", help='Can be used to evaluate multiple experiments at
           once, expected to be of kind "_{0-9}3".')
35     parser.add_argument('--spatial_data', type=str, default="msi", help='Define spatial omics data (msi or spt).')
36     return parser.parse_args()
37
```

```
38  def extract_feature_importance(classifier, feature_names, z, y, suffix, path_prefix, args):
39      top = []
40      fi_path = path_prefix + "_rf_" if args.mode == "original" else path_prefix + "/rf_"
41      writer = pd.ExcelWriter(fi_path + suffix + '.xlsx')
42      for idx, estimator in enumerate(classifier['estimator']):
43          feature_importances_ = estimator.feature_importances_
44
45          feature_importances = pd.DataFrame(feature_importances_,
46                                             index = feature_names,
47                                             columns=['importance']).sort_values('importance', ascending=False)
48
49          feature_importances.to_excel(writer, index=True, sheet_name=str(idx))
50          top.append(feature_importances[0:1])
51      writer.close()
52
53      top = pd.concat(top)
54      top_index = top.index.value_counts().nlargest(1)
55      top_importance = top["importance"].iloc[np.where(top.index == top_index.index[0])[0]]
56
57      print("Highest feature importance: ")
58      print(top_index)
59      return top_index, np.mean(top_importance)
60
61  def run_regression(exp, suffix, experiment_with_suffix, args):
62      path_prefix = os.path.join(args.ae_export, experiment_with_suffix)
63
64      random_state = random.randint(1, 1000)
65      forest = ae_rf.RandomForestRegressionBuilder(n_estimators = N_TREES, max_features = MTRY, random_state = random_state)
66
67      shuffle = ShuffleSplit(n_splits=10, random_state=42, test_size = 0.33)
68      classifier = cross_validate(forest.forest, exp.z, exp.y, scoring=args.scoring, cv=shuffle, return_estimator =True)
69
70      all_trees = []
71      for fold_estimator in classifier['estimator']:
72          individual_estimators = fold_estimator.estimators_
73          all_trees.extend(individual_estimators)
74
75      # Get the depth of the first tree
76      mean_depth = sum(tree.get_depth() for tree in all_trees) / len(all_trees)
77      min_depth = min(tree.get_depth() for tree in all_trees)
78      max_depth = max(tree.get_depth() for tree in all_trees)
79      print(f"The mean depth of all trees in the Random Forest is: {mean_depth}")
80      print(f"The min depth of all trees in the Random Forest is: {min_depth}")
81      print(f"The max depth of all trees in the Random Forest is: {max_depth}")
82      print(f"RF train score: {np.mean(classifier['test_score']) :.3f}")
83
84      feature_names = np.array(range(exp.z.shape[1]))
85      top_index, top_imp = extract_feature_importance(classifier, feature_names, exp.z, exp.y, suffix, path_prefix, args)
86
87      score = ae_rf.RandomForestSummary(suffix,
88                                        exp.z.shape[0], np.mean(classifier['test_score']),
89                                        top_index.index[0], top_imp, int(top_index.values))
90      return score
91
92
```

```
93  def run_experiments(exp, patch_adapter, suffix, args):
94      experiment_with_suffix = args.experiment + "_" + suffix
95      exp.build(n_features, args.ae_export, experiment_with_suffix).split_x_y(patch_adapter)
96
97      if args.mode == "conv_ae" or n_features == args.features:
98          exp.set_data()
99      elif args.mode == "original":
100         exp.set_reduced_data(args.features)
101
102     score = run_regression(exp, suffix, experiment_with_suffix, args)
103     return score
104
105 def main(exp, spatial, args):
106     scores = []
107     x,y = spatial.train.get_non_empty_patches()
108     print("Shape of patches", x.shape, y.shape)
109
110     patch_adapter = ae_rf.PatchAdapter(x, y, spatial._images.im_label).reduce_to_mean(args.label)
111     patch_adapter.undersample_non_label_patches(args.cutoff, args.other_fraction)
112
113     for i in range(int(args.suffix_from[1:]), int(args.suffix_to[1:]) + 1):
114         suffix = "_" + str(i).zfill(3)
115         score = run_experiments(exp, patch_adapter, suffix, args)
116         scores.append(score)
117
118     # write scores to csv
119
120 if __name__ == '__main__':
121     args = parse_args()
122     if args.mode == "conv_ae":
123         exp = msi_extension.SPTExperiment(args.experiment)
124     elif args.mode == "original":
125         exp = ae_vae.Experiment(args.experiment)
126
127     adata, _, n_features = msi_extension.read_spt_from_adata(args.h5ad_path + exp.file + ".h5ad")
128     spatial = msi_extension.SPTDataset(exp.patch_size, adata.n_vars)
129
130     spt_samples = adata[adata.obs["batch"].isin(args.samples)]
131     scaler = args.ae_export + "weights/" + args.scaler
132     spt_samples, _ = ae_preprocessing.normalize_train_test_using_scaler(spt_samples, None, scaler, spt_samples.obs, None)
133     spatial.build(spt_samples, None, create_patches = False).overlapping_patches(args.overlapping_patches)
134
135     main(exp, spatial, args)
```

Class *PatchAdapter* is used to balance the number of hypoxic and normoxic patches and to reduce each patch to a mean interpretation for input to the random forest regression model.

Code Snippet 7.11: Auxiliary classes for setting up Random Forest models

```
from sklearn.ensemble import RandomForestRegressor
import numpy as np

class PatchAdapter():
    def __init__(self, x, y, labels):
        self.x = x
        self.y = y
        self.labels = labels
        self.x_mean = None
        self.y_mean = None

    def reduce_to_mean(self, label):
        self.x_mean = np.mean(self.x, axis = (1,2))
        self.y_mean = np.mean(self.y[...,self.labels[label]], axis = (1,2))
        return self

    def undersample_non_label_patches(self, cutoff, other_fraction):
        patches_of_interest = np.where(self.y_mean >= cutoff)[0]
        len_poi = len(patches_of_interest)
        if other_fraction > 0.:
            other_patches = np.where(self.y_mean < cutoff)[0]
            other_patches = np.random.choice(other_patches, min(int(len_poi * other_fraction), len(other_patches)),
                replace=False)
        else:
            other_patches = patches_of_interest[0:1]
        print("Number of labelled patches for train/test: " + str(len_poi))
        print("Number of other patches: " + str(len(other_patches)))

        self.poi = np.union1d(patches_of_interest, other_patches)
        return self

class RandomForestSummary:
    def __init__(self, config,
                 patches, score,
                 top_name, top_imp, top_occ):
        self.config = config
        self.patches = patches
        self.score = score
        self.top_name = top_name
        self.top_imp = top_imp
        self._top_occ = top_occ

class RandomForestRegressionBuilder:
    def __init__(self, n_estimators: int, max_features, random_state):
        self.forest = RandomForestRegressor(n_estimators=n_estimators, max_features=max_features, random_state=random_state)

    def fit(self, z_train, y_train, sample_weight):
        self.forest.fit(z_train, y_train, sample_weight)
        return self
```

Code Snippet 7.12: Pre-processing of spatial transcriptomics data

```
class SPTContainer:
    MIN_COUNTS = 500
    MAX_COUNTS = 35000
    MIN_CELLS = 20

    def __init__(self, normalization: str):
        self.normalization = normalization

    def preprocess_visium_using_scanpy(self, path, count_file):
        adata = sc.read_visium(path = path, count_file = count_file)
        adata.var_names_make_unique()

        sc.pp.calculate_qc_metrics(adata, inplace=True)
        sc.pp.filter_cells(adata, min_counts=SPTContainer.MIN_COUNTS)
        sc.pp.filter_cells(adata, max_counts=SPTContainer.MAX_COUNTS)
        sc.pp.filter_genes(adata, min_cells=SPTContainer.MIN_CELLS)

        if self.normalization != "sctransform":
            sc.pp.normalize_total(adata, inplace = True)
            if self.normalization == "log":
                sc.pp.log1p(adata, copy = False)

        self.data = SPTData(0, adata)
        return self

    def build(self, norm_data, default_pixel: float):
        adata = self.data.feature

        identifier = list(adata.uns['spatial'].keys())[0]
        factor = adata.uns['spatial'][identifier]["scalefactors"]['tissue_lowres_scalef']
        if self.normalization == "sctransform":
            features = self.sort_with_sctransform(adata, norm_data, factor)
        else:
            features = self.sort(adata, factor)
        self.data = SPTData(factor, features)
        self.he = SPTImage().normalize(adata, identifier, default_pixel)
        self.fi = None

        return self

    def sort_coordinates(self, adata, factor):

        # data spots
        data_tuples = [tuple(x) for x in adata.obs[["array_row", "array_col"]].to_numpy()]
        # rows (axis 0) correspond to y
        # arrow_row = x, arrow_col = y
        # coordinates origin should be upper left corner, therefore we subtract here
        data_tuples_xy = [(x, 127 - y) for x, y in data_tuples]
        # we sort the tuples by x to match with the lowres_indices below
        data_indices = sorted(range(len(data_tuples_xy)), key=lambda i: (data_tuples_xy[i][0], data_tuples_xy[i][1]))
        data_tuples_sorted = [data_tuples_xy[i] for i in data_indices]

        # lowres spots
        indices = list(map(tuple, adata.obsm['spatial']))
        lowres_tuples = np.multiply(indices, factor)
        lowres_indices = sorted(range(len(indices)), key=lambda i: (indices[i][0], indices[i][1]))
        lowres_tuples_sorted = [(round(lowres_tuples[i][0], 2), round(lowres_tuples[i][1], 2)) for i in lowres_indices]

        obs = pd.concat([pd.DataFrame(data_tuples_sorted, columns=['x', 'y']),
                         pd.DataFrame(lowres_tuples_sorted, columns=['x_lowres', 'y_lowres'])
                        ], axis=1)

        return obs, data_indices

```

```
65      def sort_with_sctransform(self, adata, norm_data, factor):
66          var_int = norm_data.columns.intersection(adata.var.index)
67          obs_int = norm_data.index.intersection(adata.obs.index)
68          norm_data = norm_data.loc[obs_int, var_int]
69
70          obs, data_indices = self.sort_coordinates(adata, factor)
71          return ad.AnnData(norm_data.iloc[data_indices].reset_index(drop=True),
72                            # spots (obs) are only reduced in scanpy
73                            obs = obs,
74                            # some genes might have been cut in Seurat
75                            var = adata.var.loc[var_int])
76
77      def sort(self, adata, factor):
78          obs, data_indices = self.sort_coordinates(adata, factor)
79
80          return ad.AnnData(adata.X[data_indices],
81                            obs = obs,
82                            var = adata.var)
```

7.3 Results of Mass Spectrometry Imaging (Section 4.2)

Table S1: 43 peptide candidates found with at least 2 masses matched from semi-supervised convolutional autoencoder and random forest (CAERF) run to liquid chromotography (LC)-MS/MS experiment. Only one exemplary mass pair is shown, the complete data is provided in Suppl. Table 2. in Bitto et al. [111].

Protein(s)	Gene name(s)	Mass 1	Mass 2
Keratin, type II cytoskeletal 6A;Keratin, type II cytoskeletal 6C;...	KRT6A;KRT6C;KRT6B	808.387	877.441
Cytochrome b-c1 complex subunit 1, mitochondrial	UQCRC1	808.400	1042.519
Phosphoglycerate kinase 1	PGK1	808.400	1011.519
Multifunctional protein ADE2; Phosphoribosylaminoimidazole-succinocarboxamide synthase;...	PAICS	1015.501	1409.733
Prelamin-A/C;Lamin-A/C	LMNA	1042.547	971.497
Trifunctional purine biosynthetic protein adenosine-3;...	GART	1042.547	1036.548
Heterogeneous nuclear ribonucleoprotein M	HNRNPM	1425.709	822.371
Keratin, type II cytoskeletal 5	KRT5	1425.709	1424.693
60S ribosomal protein L18a	RPL18A	1425.709	1042.519
60S ribosomal protein L15;Ribosomal protein L15	RPL15	1425.709	1166.612
Cullin-associated NEDD8-dissociated protein 1	CAND1	988.480	965.469
Serine hydroxymethyltransferase, mitochondrial;...	SHMT2	988.480	854.495
Heat shock cognate 71 kDa protein	HSPA8	988.480	1409.694
DNA-dependent protein kinase catalytic subunit	PRKDC	877.466	1337.665
Eukaryotic translation initiation factor 3 subunit C;...	EIF3C;EIF3CL	1042.519	1166.637
Vigilin	HDLBP	1424.713	1061.512
Signal transducer and activator of transcription 1-alpha/beta;...	STAT1	1424.713	1307.631
Phosphoglucomutase-1	PGM1	1089.554	1026.520
V-type proton ATPase subunit B, brain isoform	ATP6V1B2	1089.554	1307.631
Keratin, type I cytoskeletal 14	KRT14	1424.693	1166.637
L-lactate dehydrogenase A chain	LDHA	1166.637	1026.520
Annexin A7	ANXA7	1043.548	1090.536
Hexokinase-1	HK1	1409.733	1030.522
Keratin, type I cytoskeletal 16	KRT16	1337.665	854.495
60S ribosomal protein L5	RPL5	1337.665	1000.473
Heterogeneous nuclear ribonucleoproteins A2/B1	HNRPA2B1;HNRNPA2B1	1337.665	1409.694
Collagen alpha-3(VI) chain	COL6A3	1036.529	989.471
Plastin-2	LCP1	997.502	1116.535
Programmed cell death protein 6	PDCD6	997.502	1338.656
Desmoplakin	DSP	1011.490	944.515
Gasdermin-A	GSDMA	944.493	1038.511

Table S1: Continued: 43 peptide candidates found with at least 2 masses matched from semi-supervised convolutional autoencoder and random forest (CAERF) run to liquid chromotography (LC)-MS/MS experiment. Only one exemplary mass pair is shown, the complete data is provided in Suppl. Table 2 in Bitto et al. [111].

Protein(s)	Gene name(s)	Mass 1	Mass 2
Basic leucine zipper and W2 domain-containing protein 1	BZW1	1061.512	807.391
Elongation factor 2	EEF2	1090.536	1038.511
Annexin A6	ANXA6	1090.536	1025.527
Protein-glutamine gamma-glutamyltransferase K	TGM1	971.497	1410.739
Eukaryotic translation initiation factor 3 subunit L	EIF3L	989.471	964.486
Galectin-7	LGALS7	909.463	856.443
T-complex protein 1 subunit eta	CCT7	909.463	944.515
Eukaryotic translation initiation factor 4 gamma 1	EIF4G1	1026.520	1410.739
Coatomer subunit alpha;Xenin;Proxenin	COPA	807.391	944.515
Plectin	PLEC	1030.522	966.486
Glyceraldehyde-3-phosphate dehydrogenase	GAPDH	1410.739	1064.539
Importin-5	IPO5	1036.548	1116.535

7.4 Results of Spatial Transcriptomics (Section 4.3)

Genes commonly found in all approaches were marked in bold.

7.4.1 Genes associated with hypoxia in exemplary unsupervised CAERF run

CSTA, ***GJB2****,* ***JUNB****, PI3,* ***TGM1****, PIEZO1, S100A9,* ***EGLN3****, C19orf33, IL36G, ABI1,* ***NFKBIA****, DLG1, SPRR1B, TNFAIP1, CDKN1A, CXCL8, KRTDAP, ACTR3, CSTB, ANKLE2, SLURP2, KRT6B, FAM110C, PDHA1, TTC14,* ***NDRG1****, S100A7, HIC2, SPRR3, FOXK1, RSRC2, SLPI, TACSTD2, MAPK1IP1L,* ***ERO1A****, PSMD10, GNA15, KDM3A, RAB2A, WNT7B, XDH, TUFT1,* ***LCN2****, TMEM154, DSG3, PPP2R2C, TRIOBP, GIPC1, ZBTB7A, MAP4K4, C14orf119, CLDN1, RAB7A, GM2A, PPP1R11, P2RY2, ITGB5, B4GALNT3, SPRR2D, IGFBP3, SCAMP4, METRNL, SERINC3, UBE2Z, YAP1, LGALSL, ADD1, PNPLA2, KCNK1,* ***GRHL1****, RNF141,* ***BNIP3L****, MXRA7, NUFIP2, ATP13A3, RAP2C, HS1BP3, PPP1R16A, BRMS1, ARAP1, CRKL, MFSD5, PSAP, ERG28, RIN2, TMEM9B, LPAR3, SIKE1, PLEC, ZFP36L1, POLD4, RSU1, RSAD1, LRRC42, SPTLC2, WAPL, CALML4, FOS, TGFB1,*

NAMPT, ATP6V0D1, CD46, SPINK7, MINK1, ANKRD13D, KYNU, CHMP2A, ZBTB7B, KLF4, ACAA2, SERINC5, ZNF185, ST14, PSME3, BTG2, ASCC2, CLDN12, BDKRB2, PLD2, MYL12B, RAB31, CD24, MAL2, ZNF839, TMEM165, LZTR1, ZBTB4, SFT2D2, UPP1, YWHAQ, MBOAT2, GOSR1, CAP1, ODC1, KCTD11, TBX6, NCOA1, R3HDM4, UBAC2, PIM1, TMEM41A, PKP3, ZNF12, ***DOCK9****,* ***MXD1****, SNX33, PLCH2, PLEKHM1, UNC50, MRFAP1L1, HEPHL1, MYO9A, CD81, POLB, RAC1, SERTAD1, IRAK1, SLC6A11, GPBP1L1, APBA3, UEVLD*

7.4.2 Genes associated with hypoxia in exemplary RF only run

FAM200B, NRIP1, ***EGLN3****,* ***LCN2****, C1QTNF12,* ***NFKBIA****, FKRP, ENKD1, CA2, BIRC3, SCAMP2, TP53BP2, MGAT5, WDR26, SNX33, SPOPL, ZC3H12C, GTDC1, FUT1,* ***ERO1A****, ISL2, FCSK, WNT3A, C6orf132, LGALS3, MPND, RNF187, ZFAND6, TTC13, TANK, TMPRSS4, GALNT3, GABARAPL2, BBS5, FBXO44, FAM83A, TMEM141, TRIM6, FBXO32, CTNNA1, ADGRF1, CCNG2, ZBTB18, IQGAP1, RYBP,* ***NDRG1****, NDUFA7, EXD3, ANGPTL4, ADM, MRVI1, SLURP1, CLDND1, PIEZO1, NUMB, PCDHGA10, ACSF3, SAP25, USP6NL,* ***GRHL1****, TCF25, NABP1, ECM1, CLDN15, PNRC1, ABLIM3, ZNRF3, BNIPL, PLPPR2, ZBTB20, NAMPT, HBP1, CGN, CASP10, ARSJ, KIAA1217, THEM4, TRPV3, FAM114A2, LCE3E, PIK3IP1, LCE3D, C11orf74,* ***JUNB****, KIAA2026, SLC5A1, SPRR2A, UBC, S100A9, S100A12, UNC13D, HILPDA, GABARAPL1, LANCL1, ACOX1, RBKS, FOSL2, SH3PXD2B, MMAA, BNIP1, EPHX3, H2AFJ,* ***GJB2****, GJB6, NFX1, ACADVL, ZBTB7B, MED19, CLDN7, MUC1, RIN2, TET1, CARHSP1, HECA, SQSTM1, ID1, FTH1, P4HA1, SASH1, TSC22D2, ANKRD37, CSNK1D, GDE1, TMEM179B, RB1, FBXO25, UGP2, RIMS3, PPP1R3B, ATG2A, NAPG, SPIN1, KLHDC7B, BHLHE40,* ***MXD1****, SLC2A1, PCBP1, DNAJC19, CHAC1, FRMD8, WSB1, PLA2G4B, RELA, LIFR, TMSB4X, OFD1, YPEL3, RAF1, DSG3,* ***DOCK9****, TMEM40, TTYH3, ITGA2,* ***BNIP3L****, MORN4, QSOX1, HK2, NIPSNAP3A, RAC1, TGOLN2, ITGAM, NRDC, WDR53, TMEM98, TCIRG1, YIPF1, HLA-E, BCKDHA, INAVA, PMAIP1, DEDD2, PLP2, LRP10, SLC20A2, ELF3, R3HDM2, MXI1, PPP1R15B, TENT2, KIFC3, NCK2,* ***TGM1****, RABGGTA, KRT16, LTB4R, DNAJC5, JUP, NIBAN2, NCOA2, YOD1, CPEB2, PICALM*

7.4.3 Genes associated with hypoxia in exemplary semi-supervised CAERF run

SPRR1B, ***TGM1****,* ***NFKBIA****, FTH1, IL1RN, ISG15,* ***EGLN3****, ANXA1, IL36G, SPRR2A,* ***GJB2****, TMPRSS11E,* ***LCN2****, OAS3, KLK10, IER5, TWF1,* ***NDRG1****, C4orf3, S100A7, FAM83A, KRT6B, UPP1, CST3, DLG1, EPHB3, RNF149, PLSCR1, TEN1, IER2, METRNL, ZC3H12A, HECTD1, SERPINB1, ANKRD11, CANT1, ELF3, TPR, PGK1,* ***JUNB****, SPRR2D, CLK1, POLD4, FOSL2, SOD2, ST5, TRIP10, FAM83G, SMAGP, MAX, PITPNC1, ACADVL, SMARCA4, PPFIA1,* ***GRHL1****, LLGL2, UBC, PGM2, ADM, EHD4, KRT80, DDIT4, SH3BP1, CKAP4, FGF11, PPIC, CDC73, MX2, RAB4A, SLC2A1, CHMP4B, CAPN15, COL17A1, JPT2, THRA, TICAM1, HEPHL1, IGFBP3, DHX15, UPK3B, TRIM21, AHNAK2, DYM, DDIT3, NEU1,* ***MXD1****, RAB5A, IRF7, CARHSP1, SERTAD1, ABLIM3, HAS3, PRRC1, SLFN5, PHC3, KRT16, GTPBP2, STOM, MARCKSL1, UBE2G1, SGK1, SEMA3C, CNFN, DIAPH1, RHOV, RSAD2, ZKSCAN1, ALG2, SERPINB6, KMT2B, STX5, STX3, RASA2, GTF2B, FBXW5, NCOA3, UBR5, BCAP31, CYB5R1, CLDN7,* ***ERO1A****, EPHA2, IL32, ENTPD4, EXOC6B, CBLC, IFNGR2, AP5B1, CDKN1A, PHRF1, USP4, TUBB6, GLTP, PCDH1, HS3ST3A1, GRB7, EPS8L2, GALNT1, GRINA, TRIM22, UBL4A, SLPI, ANO1, CEACAM1, PPP2R5D, PGLYRP3, DUSP7, GNA15, FAM162A, RAB24, ALKBH5, MUC1, C18orf25, HECA, SQSTM1, TANK, GTPBP1, ANKRD13A, CISD2, IL36RN, PERP, ERI2, ZER1, PI4K2A, FBRS, DAPK3, UNKL, SDCBP, RIDA, ZDHHC9, HK2, ARHGAP5, TUT7, RAB1A, SOCS1, ACADM, TMEM208, CALCOCO2, TPBG, RDH13, PCIF1, ALAD, LYSMD2, TBC1D10B, PICALM, RNF169, ITGB4, CERS2, PLPPR2, DDR1, SAT1, LDHA, BDH1, TBC1D22A, ANXA11, TSPO, HAUS4, TUFT1, CTDSP1, ZNF655,* ***DOCK9****, IER5L, CFB, APH1A, LMO7, MRGBP, ATP1B1, PEX13, RNF114, UBLCP1, KCTD11, ZNF592, NCEH1, TAF1D, ERP29, CHSY1, RNF213, LAPTM4A,* ***BNIP3L****, UBE2B, WFDC5, PSMD5, FAM50A, MARCH7, SNRNP200, IVL, PI3, SEC24A, ARPC5, ELL2, UBE2Q2, TRMT12, PPP2R5B, PIM1*

7.5 Combining Spatial Omics Implementation

In class *CoRegistration*, the steps for registering the aligned MSI H&E to the MSI data are achieved, by first registering an upsampled representation (method *coregister_upsampled()*), applying the transformation to the actual MSI data (method *apply()*) and performing a grid searching for the final alignment (method *___find_best_overlay()*) .

Code Snippet 7.13: Co-registration of serial spatial omics slices

```
1  import numpy as np
2  import pandas as pd
3  from scipy import stats
4  from scipy.interpolate import RBFInterpolator
5
6  class CoRegistration:
7      def __init__(self, dice_holder):
8          self._dice_holder = dice_holder
9          self._upsampled = None
10         self._result = None
11
12     @property
13     def upsampled(self):
14         if self._upsampled is None:
15             raise Exception('Data is not build yet.')
16         return self._upsampled
17
18     @property
19     def result(self):
20         if self._result is None:
21             raise Exception('Data is not build yet.')
22         return self._result
23
24     def coregister_upsampled(self, fixed: ImageRep, moving: ImageRep, deform = False):
25         moving, moving_coords = moving.crop()
26
27         # sample is resized during co-registration anyhow, so exact factor is not important
28         self._resize_factor = min(np.divide(fixed.rep.shape, moving.rep.shape))
29         moving_data_upsampled = moving.resize_by_factor(self._resize_factor)
30
31         coreg_data_upsampled, transform_params = moving_data_upsampled.co_register(fixed.rep, deform = deform)
32         self._upsampled = CoRegisteredImages(coreg_data_upsampled, 1, coreg_data_upsampled.set_mask())
33
34         scaling_factor, translate_x, translate_y = self.___extract_changing_params(transform_params.GetParameter(0,
               "TransformParameters"))
35         self._transform = CoRegistrationTransform(moving, transform_params, scaling_factor, translate_x, translate_y)
36         self._moving_coords = moving_coords
37         return self
38
39     def apply(self):
40         # coregistration of upsampled followed the scheme > scaling rotation translation
41         # applied to the data this leads to > translate rotate
42         # therefore we need to calculate the factors separately
43         padding_correction = self.___correct_for_padding(self._transform.scaling_factor)
44
45         adjusted_x = -self._transform.translate_x / self._resize_factor + padding_correction
46         adjusted_y = -self._transform.translate_y / self._resize_factor + padding_correction
47
48         self._transform = self._transform.copy(adjusted_x, adjusted_y)
49         he_factor = self._resize_factor / self._transform.scaling_factor
50         self._result = self.___find_best_overlay(he_factor)
51
52         return self
53
```

```
54
55      # rotation seems to lead to some mismatch with overlay, so we apply a grid search on the actual translation
56      def __find_best_overlay(self, he_factor):
57
58          for x_search in self._dice_holder.coord_search :
59              for y_search in self._dice_holder.coord_search :
60                  img = self._transform.apply_transform_with_adj_translate(x_search, y_search)
61
62                  spots_morphed = img.set_mask_with_shape(self.upsampled.img.rep.shape, he_factor).dilate_spots()
63                  dice_score = spots_morphed.calculate_dice_score(self.upsampled.mask)
64                  self._dice_holder.set_best_score(DiceScore(dice_score, x_search, y_search, spots_morphed))
65
66          result_img =  self._transform.apply_transform_with_adj_translate(self._dice_holder.dice.x, self._dice_holder.dice.y)
67          result_img = result_img.cropped_max()
68          return CoRegisteredImages(result_img, he_factor, self._dice_holder.dice.mask)
69
70      def __extract_changing_params(self, params):
71          scaling_factor = float(params[0])
72          translate_x = float(params[2])
73          translate_y = float(params[3])
74          return scaling_factor, translate_x, translate_y
75
76      def __correct_for_padding(self, scaling_factor):
77          adj_size = np.multiply(self._transform.img.rep.shape, self._resize_factor)
78          adj_size = np.multiply(adj_size, 1 / scaling_factor)
79
80          diff_size = np.subtract(self._upsampled.img.rep.shape, adj_size)
81          # left and right or top and bottom
82          diff_size = diff_size / 2
83          # account for smallest nessecary padding to fit dimensions
84          padding = np.max(np.floor(diff_size))
85          padding_correction =  np.ceil(padding / self._resize_factor)
86          return padding_correction
87
88      def apply_transform_to_other(self, img):
89          coords = self._moving_coords
90          cropped = img.apply_crop(coords)
91
92          transform = self._transform.copy_with_img(cropped)
93          result_img = transform.apply_transform_with_adj_translate(self._dice_holder.dice.x, self._dice_holder.dice.y)
94          result_img = result_img.apply_max_crop(self.result.img.rep.shape)
95          return result_img
96
97  class DiceScore:
98      def __init__(self, score, x, y, mask):
99          self.score = score
100         self.x = x
101         self.y = y
102         self.mask = mask
103
104 class DiceHolder:
105     def __init__(self, coord_search: list):
106         self.dice = None
107         self.coord_search = coord_search
108
109     def set_best_score(self, dice: DiceScore):
110         if self.dice is None:
111             self.dice = dice
112         elif dice.score > self.dice.score:
113             self.dice = dice
114
115
```

```
116 class CoRegisteredImages:
117     def __init__(self, img, factor, mask):
118         self.img = img
119         self.factor = factor
120         self.mask = mask
121 
122 class CoRegistrationTransform:
123 
124     def __init__(self, img, params, scaling_factor, translate_x, translate_y):
125         self.img = img
126         self.params = params
127         self.scaling_factor = scaling_factor
128         self.translate_x = translate_x
129         self.translate_y = translate_y
130 
131     def copy_with_img(self, img):
132         return CoRegistrationTransform(img, self.params, self.scaling_factor, self.translate_x, self.translate_y)
133 
134     def copy(self, translate_x, translate_y):
135         return CoRegistrationTransform(self.img, self.params, self.scaling_factor, translate_x, translate_y)
136 
137     def apply_transform_with_adj_translate(self, correct_x, correct_y):
138         transform_params = self.params
139         params = list(transform_params.GetParameter(0, "TransformParameters"))
140         params[0] = str(1)
141         params[2] = str((self.translate_x-correct_x)*-1)
142         params[3] = str((self.translate_y-correct_y)*-1)
143 
144         new_dims = self.img.rep.shape
145         transform_params.SetParameter(0, "TransformParameters", tuple(params))
146         # adjusted to match with (smaller) image
147         transform_params.SetParameter(0, 'CenterOfRotationPoint', [str(new_dims[1]/2), str(new_dims[0]/2)])
148         # set globally
149         transform_params.SetParameter('DefaultPixelValue', [str(self.img.default_pixel)])
150 
151         coreg_img = self.img.apply_transform(transform_params)
152         return coreg_img
153 
154 def interpolate_msi_spots(msi_data, spt_data, msi_im, spt_im, mz_values):
155     msi_spots_scaled =  list(map(tuple, np.multiply(msi_data.coreg.result.img.get_spots(), msi_data.coreg.result.factor )))
156     df_msi_coreg_scaled = pd.DataFrame(msi_spots_scaled, columns=['y_lowres', 'x_lowres'])
157     df_spt_scaled =  spt_data.feature.obs[['y_lowres', 'x_lowres']]
158 
159     msi_coord = np.subtract(msi_data.coreg.result.img.get_spots(), 0)
160     msi_true = [msi_im[tuple(s)] for s in msi_coord]
161 
162     interp = RBFInterpolator(df_msi_coreg_scaled, msi_true)
163     msi_interpolated = interp(df_spt_scaled)
164 
165     spt_coord = spt_data.feature.obs[['y', 'x']].to_numpy()
166     spt_true = [spt_im[tuple(s)] for s in spt_coord]
167 
168     df_msi_spots = pd.DataFrame(msi_interpolated, columns = mz_values)
169     df_spt_spots = pd.DataFrame(spt_true, columns = spt_data.feature.var_names)
170     return pd.concat([df_msi_spots, df_spt_spots], axis=1)
171 
172 
```

```
def find_correlating_spots(df_spots, n_peptides, n_genes, corr_coeff):
    df_cor_big = stats.spearmanr(df_spots)
    cor_big_statistic = df_cor_big.correlation

    upper = pd.DataFrame(cor_big_statistic, columns = df_spots.columns).where(np.triu(np.ones(cor_big_statistic.shape),
        k=1).astype(bool))
    sig = pd.DataFrame(df_cor_big.pvalue).where(np.triu(np.ones(df_cor_big.pvalue.shape), k=1).astype(bool))
    all_matches = []

    for spt_feature_idx in range(n_peptides, n_peptides + n_genes):
        high_corr = np.where(np.array(upper[upper.columns[spt_feature_idx]].values) > corr_coeff)[0]
        stat_significant = np.where(np.array(sig[sig.columns[spt_feature_idx]].values) < 0.05)[0]
        combined_condition = np.intersect1d(high_corr, stat_significant)

        found_mz_index = np.where(combined_condition < n_peptides)[0]

        if len(found_mz_index) > 0:
            all_matches.append(upper[upper.columns[spt_feature_idx]].iloc[high_corr[found_mz_index]])

    if all_matches:
        df_high_corr = pd.concat(all_matches, axis=1, keys=[s.name for s in all_matches])
    else:
        df_high_corr = pd.DataFrame()
    return df_high_corr
```

Code Snippet 7.14: Example of registration of serial spatial omics slices

```
import argparse
import glob
import re
from collections import defaultdict

import pandas as pd
import numpy as np
import cv2
import h5py

# packages created by Verena Bitto
from automsi import *
from autospt import spt_wrapper
from hemsiCNN import *

def get_params():
    sample_params = defaultdict()
    #                           (MSI Data), (MSI HE), (SPT F1)
    #                           (rotate, flip, size)
    sample_params['N150d320'] = [(250, None, None), (-10, None, 500), None]
    sample_params['N154a073'] = [(250, None, None), (-10, None, 500), None]
    sample_params['N156a074'] = [(30, None, None), (120, None, 500), None]
    sample_params['N165a002'] = [(250, None, None), (-10, None, 500), None]
    return sample_params

def parse_args():
    parser = argparse.ArgumentParser()
    parser.add_argument('--sample', type=str, help='Sample to register.')
    parser.add_argument('--msi_h5ad_path', type=str, help='Path to h5ad files')
    parser.add_argument('--he_path', type=str, help='Path to HE files.')
    parser.add_argument('--msi_files', type=str, help='Path to MSI h5ad files.')
    parser.add_argument('--spt_path', type=str, help='Path to SPT files.')
    parser.add_argument('--spt_sctransform_path', type=str, help='Path to sctransform-normalized SPT expression data, h5ad
        file.')
    parser.add_argument('--spt_normalize', type=str, default="sctransform", help='Whether or not to use sctransform
        normalization, either sctransform, total or log.')
    parser.add_argument('--corr_coeff', type=float, help='Cut of for Spearman correlation coefficient.')
    return parser.parse_args()

```

```
38  def main():
39      data =  coreg.MSIData(coreg.ConfigParams(*params[0]), train).setup_moving(default_pixel)
40      he = coreg.MovingImage(coreg.ConfigParams(*params[1])).read(msi_he_image[msi_he_index]).setup_moving(default_pixel,
            reshape = True).crop(.01).resize(cv2.INTER_AREA)
41      he.match_histograms(spt.he.fixed.rep)
42      msi = coreg.MSIContainer(data, he)
43      msi.he.co_register(spt.he.fixed.rep, deform = deform)
44
45      msi.data._coreg = coreg.CoRegistration(dice_holder).coregister_upsampled(msi.he.coreg, msi.data.moving, deform =
            deform).apply()
46
47      msi_im = msi.data.apply_transform_to_mz_values(0.0)
48      spt_im = coreg.SPTDataImage(spt.data.feature).unfold()
49      df_spots =  coreg.interpolate_msi_spots(msi.data, spt.data, msi_im, spt_im, mz_values)
50      df_high_corr = coreg.find_correlating_spots(df_spots, n_features, spt.data.feature.n_vars, args.corr_coeff)
51
52      # write df_high_corr to csv
53
54  def get_samples(path, samples):
55      return [s for s in path if any(xs in s for xs in samples)]
56
57  if __name__ == '__main__':
58      args = parse_args()
59      deform = False
60      default_pixel = 1.
61      samples = ["N150d320", "N154a073", "N156a074", "N165a002"]
62      n_number = args.sample[0]
63      m_number = args.sample[1]
64
65      msi_he_image = get_samples(glob.glob(args.he_path + "/*_HE.jpg"), samples)
66      spt_hd5 = get_samples(glob.glob(args.spt_path + "/*"), samples)
67      spt_norm = get_samples(glob.glob(args.spt_sctransform_path/*.h5"), samples)
68      file_name = "_".join([n_number, "MSI", m_number, args.spt_normalize, str(args.corr_coeff)])
69
70      spt_hd5_index = [idx for idx, s in enumerate(spt_hd5) if n_number in s][0]
71      msi_he_index = [idx for idx, s in enumerate(msi_he_image) if m_number in s][0]
72      norm_index = [idx for idx, s in enumerate(spt_norm) if n_number in s][0]
73
74      msi_adata = ad.read(args.msi_h5ad_path + args.msi_files + ".h5ad") # normalized MSI data
75      mz_values = pd.to_numeric(adata.to_df().columns, errors='coerce')
76      n_features = len(mz_values)
77
78      with h5py.File(spt_norm[norm_index], "r") as f:
79          data = f["norm_data"][:]
80          rows = np.ndarray.astype(f["rows"][:], dtype="str")
81          cols = np.ndarray.astype(f["cols"][:], dtype="str")
82      norm_data = pd.DataFrame(data, index = cols, columns = rows)
83
84      spt = spt_wrapper.SPTContainer(args.spt_normalize).preprocess_visium_using_scanpy(spt_hd5[spt_hd5_index] + "/outs/",
            "filtered_feature_bc_matrix.h5").build(norm_data, default_pixel)
85
86      dice_holder = coreg.DiceHolder(np.insert(np.arange(-4, 4, 0.5), 0, 0))
87      main()
```

Code Snippet 7.15: Auxiliary classes for spatial omics data

```
import numpy as np
import pandas as pd
import anndata as ad

class MSIContainer:
    def __init__(self, data: MSIData, he: MovingImage):
        self.data = data
        self.he = he

class DataProcessing:
    @staticmethod
    def normalize_min_max(arr, max_value):
        arr = np.asarray(arr).astype(np.float32)
        return ((arr - arr.min()) * (1/(arr.max() - arr.min()) * max_value))

    @staticmethod
    def normalize_min_max_range(arr, min_value, max_value):
        arr = np.asarray(arr).astype(np.float32)
        return ((arr - arr.min()) / (arr.max() - arr.min()) * (max_value - min_value) + min_value)

class MSIData(DataProcessing):
    def __init__(self, params, raw):
        self.params = params
        self.raw = raw
        self._img = None
        self._moving = None
        self._coreg = None

    @property
    def moving(self):
        if self._moving is None:
            raise Exception('Data is not build yet.')
        return self._moving

    @property
    def coreg(self):
        if self._coreg is None:
            raise Exception('Data is not build yet.')
        return self._coreg

    @property
    def img(self):
        if self._img is None:
            raise Exception('Data is not build yet.')
        return self._img

    def setup_moving(self, default_pixel, reshape = False):
        self._img = self.unfold_image()
        if default_pixel == 0.:
            mean_spectra = np.mean(self._img, axis = 2)
        else:
            mean_spectra = np.mean(default_pixel - self._img, axis = 2)
        mean_spectra = DataProcessing.normalize_min_max(mean_spectra, 1)
        self._moving = ImageRep(mean_spectra, default_pixel).transform(self.params)
        return self

    def unfold_image(self):
        sample = self.raw.obs[["xLocation", "yLocation"]]
        data = self.raw.to_df()
        xy_max = sample.max()
        xy_min = sample.min()
        im_range = xy_max - xy_min + 1
        im = np.full((*im_range[::-1], self.raw.shape[1]), 0.0, dtype=float)
        for i in range(sample.shape[0]):
            xy = sample.iloc[i] - xy_min
            im[tuple(xy)[::-1]] = data.iloc[i]
        return im
```

```
68
69      def apply_transform_to_mz_values(self, background):
70          n_features = self.img.shape[2]
71          msi_im = np.zeros((*self.coreg.result.img.rep.shape, n_features))
72
73          for f in range(n_features):
74              # normalization is done when converting MSI data to anndata
75              rep = np.asarray(self.img[..., f]).astype(np.float32)
76              img = ImageRep(rep, background).transform(self.params)
77              msi_im[..., f] = self.coreg.apply_transform_to_other(img).rep
78          return msi_im
79
80  class SPTDataImage:
81      def __init__(self, data: ad.AnnData):
82          self.data = data
83
84      def unfold(self):
85          sample = self.data.obs[["y", "x"]] # order matters
86          data = self.data.to_df()
87          n_features = self.data.n_vars
88          im = np.full((128, 78, n_features), 0.0, dtype=float)
89
90          for i in range(self.data.n_obs):
91              xy = sample.iloc[i]
92              im[tuple(xy)] = data.iloc[i]
93          return im
```

Code Snippet 7.16: Auxiliary class for co-registering dummy channel of H&E stain

```
1   class MovingDummyChannel():
2       def __init__(self, img):
3           self.img = img
4           self._factor = None
5           self._cropped = None
6           self._coreg = None
7           self._data_upsampled = None
8
9       @property
10      def factor(self):
11          if self._factor is None:
12              raise Exception('Factor is not set yet.')
13          return self._factor
14
15      def set_factor(self, shape):
16          self._factor = np.divide(shape, self.img.rep.shape)
17          return self
18
19      def crop(self):
20          self.img.mask.open_spots()
21          self._cropped = DimensionsFitter(self.img, self.img.mask.detect_tissue_borders()).build()
22          return self
23
24      def co_register(self, data):
25          self._cropped = self._cropped.correct_cropped_dimensions(data.rep.shape, self.img)
26          self._data_upsampled = data.resize_by_shape(self._cropped.img.rep.shape[::-1])
27          self._cropped.img.mask.close_spots(kernel_size = (4,4), iterations = 6)
28          self._coreg, _ = self._cropped.img.co_register_by_mask(self._data_upsampled)
29          return self
30
31      def resize_by_factor(self):
32          self._coreg_upsampled = self._coreg.resize_by_factor(min(self.factor))
33          return self
34
35  class DimensionsFitter:
36      def __init__(self, img, coords):
37          self.img = img
38          self.coords = coords
```

```
39
40      def build(self):
41          adj = self.img.rep[self.coords.y, self.coords.x]
42          if self.coords.is_missing():
43              adj = np.pad(adj, ((0, self.coords.missing_y), (0, self.coords.missing_x)), mode='constant',
                    constant_values=(self.img.default_pixel))
44
45          self.img = ImageRep(adj, self.img.default_pixel)
46          return self
47
48      def build_with_factor(self, level_factor):
49          self.coords = self.coords.multiply(level_factor)
50          return self.build()
51
52      def correct_cropped_dimensions(self, data_shape, he_original):
53          missing_y, missing_x = 0, 0
54          data_y, data_x = data_shape
55          actual_y, actual_x = self.img.rep.shape
56          aspect_ratio = np.divide(data_y, data_x)
57          print("Aspect ratio of data", aspect_ratio)
58
59          expected_x = np.divide(actual_y, aspect_ratio)
60          expected_y = np.multiply(actual_x, aspect_ratio)
61
62          if expected_x > actual_x: # we cropped too many pixels at x dimensions
63              coords_adj = self.coords.crop_to_fit_ratio(actual_y, int(np.round(expected_x)))
64              coords_adj.set_missing_x(coords_adj.x.stop - he_original.rep.shape[1])
65
66          elif expected_y > actual_y: # we cropped too many pixels at y dimensions
67              coords_adj = self.coords.crop_to_fit_ratio(int(np.round(expected_y)), actual_x)
68              coords_adj.set_missing_y(coords_adj.y.stop - he_original.rep.shape[0])
69
70          he_cropped_adj = DimensionsFitter(he_original, coords_adj).build()
71          return he_cropped_adj
72
73  class DimensionsSlice:
74      def __init__(self, y, x):
75          self.y = y
76          self.x = x
77          self.missing_y = 0
78          self.missing_x = 0
79
80      def set_missing_y(self, y):
81          self.missing_y = y
82
83      def set_missing_x(self, x):
84          self.missing_x = x
85
86      def is_missing(self):
87          return self.missing_x > 0 or self.missing_y > 0
88
89      def crop_to_fit_ratio(self, y_dim, x_dim):
90          y_diff = int(self.y.stop - self.y.start - y_dim)
91          x_diff = int(self.x.stop - self.x.start - x_dim)
92          return DimensionsSlice(slice(self.y.start, self.y.stop + y_diff * -1), slice(self.x.start, self.x.stop + x_diff *
                -1))
93
94      def _multiply(self, dim, level_factor):
95          dim_multiplied = np.multiply((dim.start, dim.stop), level_factor)
96          return slice(np.floor(dim_multiplied[0]).astype(int), np.ceil(dim_multiplied[1]).astype(int))
97
98      def multiply(self, level_factor):
99          y = self._multiply(self.y, level_factor[0])
100         x = self._multiply(self.x, level_factor[1])
101         coords = DimensionsSlice(y, x)
102         coords.set_missing_y(np.round(self.missing_y * level_factor[0]).astype(int))
103         coords.set_missing_x(np.round(self.missing_x * level_factor[1]).astype(int))
104         return coords
```

Code Snippet 7.17: Auxiliary classes for processing high-resolution HE images

```
class MovingRGBImage():
    def __init__(self, params: ConfigParams):
        self.params = params
        self.channel_r = None # MovingImage
        self.channel_g = None # MovingImage
        self.channel_b = None # MovingImage
        self.moving = None

    def read(self, path, level):
        with tifff.TiffFile(path) as tif:
            self.original = tif.series[0].levels[level].asarray()
        return self

    def get_shape(self):
        return self.channel_r.moving.rep.shape

    def setup_moving(self, default_pixel, reshape = False):
        self.channel_r = MovingImage(self.params).set_original(self.original[...,0]).setup_moving(default_pixel, reshape)
        self.channel_g = MovingImage(self.params).set_original(self.original[...,1]).setup_moving(default_pixel, reshape)
        self.channel_b = MovingImage(self.params).set_original(self.original[...,2]).setup_moving(default_pixel, reshape)

        self.original = None
        return self

    def crop(self, threshold: float = 0.001):
        self.channel_r = self.channel_r.crop(threshold)
        self.channel_g = self.channel_g.apply_crop(self.channel_r._coords)
        self.channel_b = self.channel_b.apply_crop(self.channel_r._coords)
        return self

    def rotate(self, angle, reshape):
        self.channel_r = self.channel_r.rotate(angle, reshape)
        self.channel_g = self.channel_g.rotate(angle, reshape)
        self.channel_b = self.channel_b.rotate(angle, reshape)
        return self

    def crop_by_example(self, dummy):
        self.channel_r = self.channel_r.crop_by_example(dummy._cropped.coords, dummy.factor)
        self.channel_g = self.channel_g.crop_by_example(dummy._cropped.coords, dummy.factor)
        self.channel_b = self.channel_b.crop_by_example(dummy._cropped.coords, dummy.factor)
        return self

    def co_register(self, template):
        transform_map = self.channel_r.co_register(template.rep)
        self.channel_g = self.channel_g.apply_transform(transform_map)
        self.channel_b = self.channel_b.apply_transform(transform_map)
        return self

    def downsize(self, factor):
        self.channel_r = self.channel_r.downsize(factor)
        self.channel_g = self.channel_g.downsize(factor)
        self.channel_b = self.channel_b.downsize(factor)

        return self

    def pad(self, padding):
        self.channel_r = self.channel_r.pad(padding)
        self.channel_g = self.channel_g.pad(padding)
        self.channel_b = self.channel_b.pad(padding)
        return self

    def stack_channels(self):
        channels = np.stack([self.channel_r.coreg.rep, self.channel_g.coreg.rep, self.channel_b.coreg.rep])
        channels = np.moveaxis(channels, 0, -1)
        return channels
```

Code Snippet 7.18: Auxiliary classes for creating patches from data and image

```
class ImagePatchBuilder:

    def __init__(self, img, img_patch_size, data_patch_size):
        self.original = img
        self.img_patch_size = img_patch_size
        self.data_patch_size = data_patch_size
        self.img = None
        self.error = 0

    def derive_params(self, data_shape):
        data_size = self.data_patch_size * np.ceil(max(data_shape) / self.data_patch_size).astype(int)
        data_padding = np.subtract(data_size, data_shape)
        no_patches = int(data_size * data_size / (self.data_patch_size * self.data_patch_size))

        img_size = int(np.sqrt(self.img_patch_size * self.img_patch_size * no_patches))
        img_padding = np.multiply(data_padding, np.divide(self.img_patch_size, self.data_patch_size))
        expected_padding = img_padding[np.argmax(self.original.rep.shape)]
        # assuming the actual image size is larger
        downsample_factor = min(np.divide(img_size - expected_padding, self.original.rep.shape))

        img_shape = np.multiply(self.original.rep.shape, downsample_factor)
        img_shape = (round(img_shape[0]), round(img_shape[1]))

        img_size = self.img_patch_size * np.ceil(max(img_shape) / self.img_patch_size).astype(int)
        padding = np.subtract(img_size, img_shape)

        he_data_ratio = np.divide(img_shape, data_shape)
        data_padding_expected = np.multiply(data_padding, he_data_ratio)
        error = np.subtract(data_padding_expected, padding)

        return no_patches, downsample_factor, padding, error
```

For co-registration of H&E image and MSI data, the parameters (coords, padding) are calculated on the level 2 H&E image and applied to level 0 H&E image. Code Snippet 7.19 also highlights the creation of patches for subsequent training of a CNN.

Code Snippet 7.19: Example for registration and patch creation of high-resolution H&E images and data

```
import sys
import argparse

import glob
import re
from collections import defaultdict
import numpy as np
import pandas as pd
import cv2

import tifffile as tifff
import matplotlib.pyplot as plt
import matplotlib.colors as colors
import tensorflow as tf

# packages created by Verena Bitto
from automsi import ae_preprocessing, ae_images
from hemsiCNN import coreg, hemsi_datasets

```

```
20  def parse_args():
21      parser = argparse.ArgumentParser()
22      parser.add_argument('--msi_h5ad_path', type=str, help='Path to h5ad file')
23      parser.add_argument('--msi_files', type=str, help='Path to h5ad file')
24      parser.add_argument('--msi_hires_he_import', type=str, help='HE stain from MSI data')
25      parser.add_argument('--analysis_export', type=str, help='Location were tfrecord are stored')
26      parser.add_argument('--sample', type=str, help='Unique identifier for sample')
27      parser.add_argument('--msi_hires_level', type=int, help='TIFF level (0-2)')
28      parser.add_argument('--he_stride', type=int, help='Stride for overlapping patches (120, 240, 360)')
29      parser.add_argument('--mode', type=str, help='Write all patches (all) or sample for hypoxia (hypoxia)')
30      parser.add_argument('--fi_cutoff', type=float, help='Cutoff for FI annotations')
31      parser.add_argument('--other_fraction', type=float, default=1, help='Defines the fraction of non-hypoxic patches to be
            sampled relative to the number of hypoxic patches.')
32      parser.add_argument('--suffix', type=str, help='Only used for saving file name')
33      return parser.parse_args()
34
35  def get_params():
36      sample_params = defaultdict()
37      # We need to flip images according to SPT data
38      sample_params['_M825'] = [(180, None, None)] # N154b184
39      sample_params['_M815'] = [(150, None, None)] # N154a037
40      sample_params['_M819'] = [(120, None, None)] # N154a073
41      sample_params['_M821'] = [(60, None, None)] # N154a098
42      return sample_params
43
44  def create_patches(patch_size, n_features, im, samples, stride):
45      patches_ = ae_images.Patches(patch_size, n_features)
46      patches = patches_.create_all_with_stride(pd.Series([im], index=samples), stride)
47      return patches_, np.concatenate(patches)
48
49  def setup_l2_he(he):
50      he_l2_r = coreg.MovingDummyChannel(he_l2.channel_r.moving).crop().co_register(data.moving)
51      he_l2_r = he_l2_r.set_factor(he.get_shape())
52      # upscale coreg image to match roughly he
53      he_l2_r = he_l2_r.resize_by_factor()
54
55      return he_l2_r
56
57
```

```
58  def setup_l0(args, he, he_l2_r):
59      he.crop_by_example(he_l2_r)
60      he = he.co_register(he_l2_r._coreg_upsampled)
61
62      no_patches, downsample_factor, padding, error = coreg.ImagePatchBuilder(he.channel_r._coreg, he_patch_size,
            data_patch_size).derive_params(data.moving.rep.shape)
63
64      he = he.downsize(downsample_factor).pad(padding)
65      he_channels = he.stack_channels()
66      plot_patches(he_channels, no_patches, "HE")
67      he_patches_, he_patches_all = create_patches(he_patch_size, 3, he_channels, [m_number], args.he_stride)
68
69      return he_patches_all
70
71  def main(args):
72      he = coreg.MovingRGBImage(coreg.ConfigParams(*params[0])).read(msi_hires_he_import,
            args.msi_hires_level).setup_moving(default_pixel, reshape = True)
73
74      he_l2_r = setup_l2_he(he)
75      he_patches = setup_l0(args, he, he_l2_r)
76
77      msi_patches =  msi_dataset.train.x.flatten()
78      msi_patches = np.asarray(msi_patches,  dtype=np.float32)
79
80      non_zero_p = msi_dataset.train.x.identify_non_zero_patches()
81      patch_mean = np.mean(he_patches, axis = (1,2,3))
82      non_zero_he = np.where(patch_mean < .95)[0]
83
84      print("Non background patches", len(non_zero_p))
85      print("Non background HE patches", len(non_zero_he))
86      non_zero_intersect = np.intersect1d(non_zero_he, non_zero_p)
87
88      # write he_patches[non_zero_intersect], msi_patches[non_zero_intersect] to TFRecordDataset
89
90
91  if __name__ == '__main__':
92      args = parse_args()
93      data_patch_size = 3
94      he_patch_size = 360
95      default_pixel = 1.
96
97      msi_hires_he_import = glob.glob(args.msi_hires_he_import + '*.ome.tif')
98      msi_hires_he_import = [s for s in msi_hires_he_import if args.sample in s][0]
99
100     m_number = re.search("_M[0-9]{3}", msi_hires_he_import).group(0)
101     tumor_model = re.search("N[0-9]{3}", msi_hires_he_import).group(0)
102     params = get_params()[m_number]
103     print("start processing... " + m_number)
104
105     adata = ad.read(args.msi_h5ad_path + args.msi_files + ".h5ad") # normalized MSI data
106     mz_values = pd.to_numeric(adata.to_df().columns, errors='coerce')
107     n_features = len(mz_values)
108
109     data =  coreg.MSIData(coreg.MovingImgParams(None), train).setup_moving(default_pixel)
110     he_l2 = coreg.MovingRGBImage(coreg.ConfigParams(*params[0])).read(msi_hires_he_import, 2).setup_moving(default_pixel,
            reshape = True)
111
112     data_stride = int(data_patch_size / (he_patch_size / args.he_stride))
113     msi_dataset = ae_images.MSIDataset(data_patch_size, n_features, im_label = {'FI': 0}, obs_label = {"FI" :
            "fi"}).build(train, None, create_patches = True).overlapping_patches(data_stride)
114
115     naming = [m_number, "level", args.msi_hires_level, args.he_stride, args.fi_cutoff, args.other_fraction, args.mode]
116     prefix = '_'.join(str(n) for n in naming)
117
118     main(args)
```

Glossary

AE An autoencoder (AE) can be used as (non-linear) dimensionality reduction method and is based on two neural networks (encoder / decoder) to compress data. xv, 3, 20, 21, 32, 33, 36, 37, 38, 39, 40, 42, 45, 47, 52, 60, 68, 77, 78, 79, 81, 83

CAE A convolutional autoencoder (CAE) is a specific type of autoencoder (CAE) which incorporates convolutional layers to learn the spatial context. 2, 3, 4, 28, 33, 38, 42, 46, 66, 67, 68, 76, 77, 82, 83, 96, 98, 102, 104, 110, 111

CAERF CAERF in this thesis describes the abbreviation for the proposed combined convolutional autoencoder approach which is based on dimensionality reduction of spatial omics data using convolutional autoencoders and subsequent analysis using a RF regression model. 2, 3, 4, 33, 40, 42, 48, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 68, 69, 70, 71, 72, 73, 74, 75, 76, 80, 81, 82, 83, 92, 93, 98, 99, 113, 146, 147

CI A confidence interval (CI) is a statistical estimate of a parameter along with a specified level of confidence. 62

CNN A convolutional neural network (CNN) is a machine learning algorithm which tries to derive and learn patterns from images using so-called convolutional layers. 4, 27, 28, 76, 78, 95, 96, 98, 99, 101, 102, 103, 104, 106, 107, 108, 109, 110, 111, 159

DAPI 4',6-diamidino-2-phenylindole (DAPI) is commonly used to label the nuclei of cells. 28

DICE The DICE score is a measure of spatial overlap of two binary images. 90

DSF Disease-free survival is a clinical endpoint which is a defined time interval in which patient does not face any signs or symptoms of cancer. Exact definition may differ per study. Also termed relapse-free survival or recurrence-free survival (RFS) [13]. 5

EFS Event-free survival is a clinical endpoint which is a defined time interval in which a patient is free of a dedicated event, e.g., disease progression. Defined events may differ per study. [13]. 5, 6

ESI Electrospray ionization (ESI) is an ionization technique in mass spectrometry. 7

FAZA ^{18}F-fluoroazomycin-arabinoside (FAZA) is a radiotracer used in PET imaging. 10

FDR The false discovery rate (FDR) is a statistical measure representing the proportion of false positives among all declared positive results. 11, 48

FFPE Formalin-fixed paraffin-embedding (FFPE) is a common method to preserve tissue through formalin. 26

FI A fluorescence image (FI) depicts fluorescent dyes to illuminate intracellular molecules [181]. 25, 28, 34, 35, 64, 69, 80, 82, 106, 109, 117

FMISO ^{18}F-fluoromisonidazole (FMISO) is a radiotracer used in PET imaging. 10

FTICR Fourier Transform Ion Cyclotron Resonance (FTICR) is a type of mass analyzer used in mass spectrometry. Masses are measured by the cyclotron motion frequency of ions within a magnetic field. 8, 80

FWHM In MS, full width at half maximum (FHWM) represents the peak width at half of its maximum intensity. A smaller FWHM indicates higher resolution and precision. 49, 50

H&E Hematoxylin and eosin (H&E) is commonly used to stain tissue slices. Hematoxylin stains cell nuclei blue, while eosin stains the cytoplasm pink. H&E stains allow to detect specific structures in cells as well as cancerous cells. [181]. 4, 8, 9, 21, 25, 27, 28, 63, 64, 65, 80, 81, 85, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 115, 117, 150, 159

HNSCC Head and neck squamous cell carcinoma (HNSCC) is a collective term for cancers which arise in the hypopharynx, larynx, nasopharynx, oral cavity and oropharynx. 1, 2, 3, 4, 5, 6, 9, 10, 11, 23, 27, 32, 52, 55, 76, 79, 81, 82, 85, 102, 106, 113, 114, 115

HPV Human papillomavirus (HPV) is a known risk factor for the development of HNSCC [181]. 6, 24

IBI Impurity-based importance (IBI) gauges the importance of a feature using a defined impurity measure, i.e., Gini index for classification or MSE for regression tasks. 13, 48, 76, 77

LC-MS/MS Liquid chromotography (LC) combines liquid chromatography with tandem mass spectrometry to identify compounds in a tissue. 3, 8, 26, 29, 43, 44, 45, 48, 49, 50, 54, 55, 56, 58, 62, 76, 79, 92, 146, 147

LRC Loco-regional tumor control (LRC) is a clinical endpoint which is a defined time interval in which no further tumor growth was observed for the primary tumor. Exact definition may differ per study [13], see also LRF. 6, 9, 10, 11

LRF Loco-regional tumor failure (LRF) is a clinical endpoint, defined as the time from randomization to the first loco-regional relapse, see also LRC. 5, 6

m/z The mass-to-charge ratio (m/z) is the mass number of an ion divided by its charge number, used in mass spectrometry. xvii, 3, 4, 7, 8, 26, 43, 44, 45, 46, 47, 48, 49, 53, 54, 56, 57, 58, 59, 60, 61, 62, 63, 77, 79, 90, 98, 99, 100, 101, 102, 103, 104, 106, 109

MAE The mean absolute error (MAE) is a commonly used loss function to quantify the average absolute difference e.g., between the input and the reconstructed output of autoencoders. 18, 21, 42

MALDI Matrix-assisted laser desorption ionization (MALDI) is an ionization technique which incorporates a matrix during mass spectrometry. 7, 8, 26, 48, 49, 76

MI Mutual information (MI) is a common metric used (among others) in image co-registration. It quantifies the degree of shared information between two sources. 22, 35, 97, 110

MS Mass spectrometry (MS) measures masses of charged mass analytes. 3, 7, 8, 43, 49, 78, 79, 80

MSE The mean squared error (MSE) is a commonly used loss function to quantify the average squared difference e.g., between the input and the reconstructed output of autoencoders. 13, 17, 18, 21, 48, 77

MSI Mass spectrometry imaging (MSI) measures masses of charged mass analytes in a spatially resolved context. 1, 2, 3, 4, 8, 16, 23, 25, 26, 27, 28, 29, 31, 32, 41, 43, 44, 48, 49, 50, 51, 52, 53, 54, 63, 68, 76, 77, 78, 79, 81, 82, 83, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 104, 105, 106, 108, 109, 110, 113, 114, 115, 140, 150, 159

NN A neural network (NN) is a machine learning algorithm which tries to derive and learn patterns from data (see CNN). 17, 18, 19, 20, 21, 81

OS Overall survival (OS) is considered as the gold standard clinical endpoint for cancer. It is defined as time from randomization to any death [11]. In clinical trails, usually only a dedicated period of time is considered, e.g., five years after treatment [181]. 5, 9

PC A principal component (PC) in context of PCA is a linear combination of the original features, maximizing variance. 15

PCA Principal componeent analysis (PCA) is a linear dimensionality reduction method. 15, 16, 17, 18, 20

PET Positron emission tomography (PET) is a molecular imaging technique with different tracer allowing to visualize biological processes [41]. 8, 10

PI Permutation importance (PI) estimates a feature's contribution to a model by shuffling its values and calculating the resulting performance loss. 13, 76, 77

pimonidazole Anti-pimonidazole polyclonal antibody (pimonidazole) can be utilized to recognize pimonidazole adducts as a biochemical marker of hypoxic cells. 27, 28, 52, 79

ppm Parts per million (ppm) is a unit denoting one part in a million and is commonly used to express errors in MS measurements. 49

RCTx Primary radiochemotherapy (RCTx) is a combinational therapy of radiotherapy and chemotherapy. In Germany, it is to date the standard treatment for patients with inoperable advanced HNSCC. 5, 6, 9, 10, 11, 23, 24

ReLU A rectified linear unit (ReLU) is a commonly used activation function in neural networks. For a given input, it outputs positive values directly and zero for negative values, thereby introducing non-linearity. 17, 19, 37, 126

RF A random forest is a commonly applied feature selection method based on the predictions of multiple decisions trees. 2, 4, 13, 14, 28, 32, 33, 40, 47, 48, 52, 53, 54, 56, 57, 58, 59, 60, 61, 62, 63, 68, 69, 70, 71, 72, 73, 74, 76, 78, 79, 82, 83

RNA Ribonucleic acid (RNA) is a nuclei acid present in every cell. One can distinugish different types of RNA: messenger RNA (mRNA), ribosomal RNA (rRNA) and transfer RNA (tRNA). 7

scRNA Single cell RNA sequencing (scRNA) captures the transcriptome of individual cells (different to bulk RNA sequencing). 7, 17, 78

SNR The signal-to-noise ratio (SNR) assesses the ratio of meaningful signal of data to background noise. 44

SPT Spatial transcriptomics measures gene expression profiles in a spatially resolved context. 1, 2, 3, 4, 23, 25, 26, 27, 28, 29, 32, 63, 64, 65, 66, 68, 69, 70, 76, 78, 79, 81, 82, 83, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 105, 107, 108, 109, 110, 113, 114, 140

SSD The sum of squared differences (SSD) is a common metric used (among others) in image co-registration. It measures the total sum of squared differences (like pixel intensity) between two sources. 22

SSIM The structural similarity index measure (SSIM) is a metric to quantify the similarity between images, whereas 1 denotes a perfect similarity and 0 indicates complete dissimilarity. 58, 59, 60, 61, 73, 74, 75, 81, 82

TIC Total ion current (TIC) or total ion count normalization adjusts for variations in MS intensities by scaling individual spectra by their respective total ion currents, i.e., the overall signal intensities [182]. 45

t-SNE T-distributed stochastic neighbor embedding (t-SNE) is a non-linear dimensionality reduction method. 16, 17

TCR Tumor control rate (TCR) evaluates the percentage of controlled tumors, i.e., tumors which either shrunk in size or remained stable over a specific period of time after treatment. 11

TOF Time of flight (TOF) is a commonly used mass analyzer in mass spectrometry. Masses are measured by the time an object needs to travel a given distance. 8, 26, 77, 80

UMAP Uniform manifold approximation and projection (UMAP) is a non-linear dimensionality reduction method. 16, 17

References

1. Harris AL. Hypoxia — a key regulatory factor in tumour growth. *Nat Rev Cancer* **2**, 38–47 (2002).
2. Thiruthaneeswaran N, Bibby BA, Yang L, Hoskin PJ, Bristow RG, Choudhury A, and West C. Lost in application: Measuring hypoxia for radiotherapy optimisation. *European Journal of Cancer* **148**, 260–276 (2021).
3. Yang L and West CM. Hypoxia gene expression signatures as predictive biomarkers for personalising radiotherapy. *The British Journal of Radiology* 20180036 (2018).
4. Ranzato M, Huang FJ, Boureau YL, and LeCun Y. Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, (2007). DOI: 10.1109/cvpr.2007.383157.
5. Najjar R. Redefining Radiology: A Review of Artificial Intelligence Integration in Medical Imaging. *Diagnostics* **13**, 2760 (2023).
6. Zou J, Huss M, Abid A, Mohammadi P, Torkamani A, and Telenti A. A primer on deep learning in genomics. *Nature Genetics* **51**, 12–18 (2018).
7. Ferlay J, Colombet M, Soerjomataram I, Parkin DM, Piñeros M, Znaor A, and Bray F. Cancer statistics for the year 2020: An overview. *International Journal of Cancer*. (2021).
8. Linge A, Lohaus F, Löck S, Nowak A, Gudziol V, Valentini C, Neubeck C von, Jütz M, Tinhofer I, Budach V, et al. HPV status, cancer stem cell marker expression, hypoxia gene signatures and tumour volume identify good prognosis subgroups in patients with HNSCC after primary radiochemotherapy: A multicentre retrospective study of the German Cancer Consortium Radiation Oncology Group (DKTK-ROG). *Radiotherapy and Oncology* **121**, 364–373 (2016).

9. Pignon J, Bourhis J, Domenge C, and Designé L. Chemotherapy added to locoregional treatment for head and neck squamous-cell carcinoma: three meta-analyses of updated individual data. *The Lancet* **355**, 949–955 (2000).
10. Pignon JP, Maître A le, Maillard E, and Bourhis J. Meta-analysis of chemotherapy in head and neck cancer (MACH-NC): An update on 93 randomised trials and 17,346 patients. *Radiotherapy and Oncology* **92**, 4–14 (2009).
11. Pazdur R. Endpoints for Assessing Drug Activity in Clinical Trials. *The Oncologist* **13**, 19–21 (2008).
12. McKee AE, Farrell AT, Pazdur R, and Woodcock J. The Role of the U.S. Food and Drug Administration Review Process: Clinical Trial Endpoints in Oncology. *The Oncologist* **15**, 13–18 (2010).
13. Hezel M, Usslar K von, Kurzweg T, Lörincz BB, and Knecht R. Endpoints and cutpoints in head and neck oncology trials: methodical background, challenges, current practice and perspectives. *European Archives of Oto-Rhino-Laryngology* **273**, 837–844 (2015).
14. Lacas B, Carmel A, Landais C, Wong SJ, Licitra L, Tobias JS, Burtness B, Ghi MG, Cohen EE, Grau C, et al. Meta-analysis of chemotherapy in head and neck cancer (MACH-NC): An update on 107 randomized trials and 19,805 patients, on behalf of MACH-NC Group. *Radiotherapy and Oncology* **156**, 281–293 (2021).
15. Dayyani F, Etzel CJ, Liu M, Ho CH, Lippman SM, and Tsao AS. Meta-analysis of the impact of human papillomavirus (HPV) on cancer risk and overall survival in head and neck squamous cell carcinomas (HNSCC). *Head Neck Oncol* **2**. (2010).
16. Albers AE, Qian X, Kaufmann AM, and Coordes A. Meta analysis: HPV and p16 pattern determines survival in patients with HNSCC and identifies potential new biologic subtype. *Scientific Reports* **7**. (2017).
17. Lassen P, Eriksen JG, Hamilton-Dutoit S, Tramm T, Alsner J, and Overgaard J. HPV-associated p16-expression and response to hypoxic modification of radiotherapy in head and neck cancer. *Radiotherapy and Oncology* **94**, 30–35 (2010).
18. Krause M, Alsner J, Linge A, Bütof R, Löck S, and Bristow R. Specific requirements for translation of biological research into clinical radiation oncology. *Molecular Oncology* **14**, 1569–1576 (2020).
19. Janssen HL, Haustermans KM, Balm AJ, and Begg AC. Hypoxia in head and neck cancer: How much, how important? *Head Neck* **27**, 622–638 (2005).

20. Overgaard J. Hypoxic modification of radiotherapy in squamous cell carcinoma of the head and neck – A systematic review and meta-analysis. *Radiother. Oncol.* **100**, 22–32 (2011).
21. Sarhadi VK and Armengol G. Molecular Biomarkers in Cancer. *Biomolecules* **12**, 1021 (2022).
22. Abdolahi S, Ghazvinian Z, Muhammadnejad S, Saleh M, Asadzadeh Aghdaei H, and Baghaei K. Patient-derived xenograft (PDX) models, applications and challenges in cancer research. *Journal of Translational Medicine* **20**. (2022).
23. Wang Z, Gerstein M, and Snyder M. RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics* **10**, 57–63 (2009).
24. Tellez-Gabriel M, Ory B, Lamoureux F, Heymann MF, and Heymann D. Tumour Heterogeneity: The Key Advantages of Single-Cell Analysis. *International Journal of Molecular Sciences* **17**, 2142 (2016).
25. Marx V. Method of the Year: spatially resolved transcriptomics. *Nature Methods* **18**, 9–14 (2021).
26. Sounart H, Lázár E, Masarapu Y, Wu J, Várkonyi T, Glasz T, Kiss A, Borgström E, Hill A, Rezene S, et al. Dual spatially resolved transcriptomics for human host–pathogen colocalization studies in FFPE tissue sections. *Genome Biology* **24**. (2023).
27. Lewis SM, Asselin-Labat ML, Nguyen Q, Berthelet J, Tan X, Wimmer VC, Merino D, Rogers KL, and Naik SH. Spatial omics and multiplexed imaging to explore cancer biology. *Nature Methods* **18**, 997–1012 (2021).
28. Aebersold R, Anderson L, Caprioli R, Druker B, Hartwell L, and Smith R. Perspective: A Program to Improve Protein Biomarker Discovery for Cancer. *Journal of Proteome Research* **4**, 1104–1109 (2005).
29. Matthiesen R and Mutenda KE. Introduction to Proteomics. In: *Mass Spectrometry Data Analysis in Proteomics*. Methods in Molecular Biology. (2007):1–36. DOI: 10.1385/1-59745-275-0:1.
30. Patterson SD and Aebersold RH. Proteomics: the first decade and beyond. *Nature Genetics* **33**, 311–323 (2003).
31. Zhu X, Xu T, Peng C, and Wu S. Advances in MALDI Mass Spectrometry Imaging Single Cell and Tissues. *Frontiers in Chemistry* **9**. (2022).

32. Jungmann JH and Heeren RM. Emerging technologies in mass spectrometry imaging. *Journal of Proteomics* **75**, 5077–5092 (2012).
33. Spraggins JM, Rizzo DG, Moore JL, Rose KL, Hammer ND, Skaar EP, and Caprioli RM. MALDI FTICR IMS of Intact Proteins: Using Mass Accuracy to Link Protein Images with Proteomics Data. *Journal of The American Society for Mass Spectrometry* **26**, 974–985 (2015).
34. Spraggins JM, Rizzo DG, Moore JL, Noto MJ, Skaar EP, and Caprioli RM. Next-generation technologies for spatial proteomics: Integrating ultra-high speed MALDI-TOF and high mass resolution MALDI FTICR imaging mass spectrometry for protein analysis. *PROTEOMICS* **16**, 1678–1689 (2016).
35. Taylor AJ, Dexter A, and Bunch J. Exploring Ion Suppression in Mass Spectrometry Imaging of a Heterogeneous Tissue. *Analytical Chemistry* **90**, 5637–5645 (2023).
36. Römpp A and Spengler B. Mass spectrometry imaging with high resolution in mass and space. *Histochemistry and Cell Biology* **139**, 759–783 (2013).
37. Palmblad M. Retention Time Prediction and Protein Identification. In: *Mass Spectrometry Data Analysis in Proteomics*. Humana Press, (2007):195–208. DOI: 10.1385/1-59745-275-0:195.
38. Bemis KD, Harry A, Eberlin LS, Ferreira CR, Ven SM van de, Mallick P, Stolowitz M, and Vitek O. Probabilistic Segmentation of Mass Spectrometry (MS) Images Helps Select Important Ions and Characterize Confidence in the Resulting Segments. *Molecular & Cellular Proteomics* **15**, 1761–1772 (2016).
39. Buchberger AR, DeLaney K, Johnson J, and Li L. Mass Spectrometry Imaging: A Review of Emerging Advancements and Future Insights. *Analytical Chemistry* **90**, 240–265 (2017).
40. Ščupáková K, Balluff B, Tressler C, Adelaja T, Heeren RM, Glunde K, and Ertaylan G. Cellular resolution in clinical MALDI mass spectrometry imaging: the latest advancements and current challenges. *Clinical Chemistry and Laboratory Medicine (CCLM)* **58**, 914–929 (2019).
41. Pantel AR and Mankoff DA. Molecular imaging to guide systemic cancer therapy: Illustrative examples of PET imaging cancer biomarkers. *Cancer Letters* **387**, 25–31 (2017).
42. Chan JKC. The Wonderful Colors of the Hematoxylin–Eosin Stain in Diagnostic Surgical Pathology. *International Journal of Surgical Pathology* **22**, 12–32 (2014).

43. Winter SC, Buffa FM, Silva P, Miller C, Valentine HR, Turley H, Shah KA, Cox GJ, Corbridge RJ, Homer JJ, et al. Relation of a Hypoxia Metagene Derived from Head and Neck Cancer to Prognosis of Multiple Cancers. *Cancer Research* **67**, 3441–3449 (2007).
44. Sørensen BS, Toustrup K, Horsman MR, Overgaard J, and Alsner J. Identifying pH independent hypoxia induced genes in human squamous cell carcinomasin vitro. *Acta Oncologica* **49**, 895–905 (2010).
45. Buffa FM, Harris AL, West CM, and Miller CJ. Large meta-analysis of multiple cancers reveals a common, compact and highly prognostic hypoxia metagene. *British Journal of Cancer* **102**, 428–435 (2010).
46. Toustrup K, Sørensen BS, Nordsmark M, Busk M, Wiuf C, Alsner J, and Overgaard J. Development of a Hypoxia Gene Expression Classifier with Predictive Impact for Hypoxic Modification of Radiotherapy in Head and Neck Cancer. *Cancer Research* **71**, 5923–5931 (2011).
47. Eustace A, Mani N, Span PN, Irlam JJ, Taylor J, Betts GN, Denley H, Miller CJ, Homer JJ, Rojas AM, et al. A 26-Gene Hypoxia Signature Predicts Benefit from Hypoxia-Modifying Therapy in Laryngeal Cancer but Not Bladder Cancer. *Clinical Cancer Research* **19**, 4879–4888 (2013).
48. Harris B, Barberis A, West C, and Buffa F. Gene Expression Signatures as Biomarkers of Tumour Hypoxia. *Clinical Oncology* **27**, 547–560 (2015).
49. Deschuymer S, Sørensen BS, Dok R, Laenen A, Hauben E, Overgaard J, and Nuyts S. Prognostic value of a 15-gene hypoxia classifier in oropharyngeal cancer treated with accelerated chemoradiotherapy. *Strahlenther. Onkol.* **196**, 552–560 (2020).
50. Grégoire V, Tao Y, Kaanders J, Machiels J, Vulquin N, Nuyts S, Fortpied C, Lmalem H, Marreaud S, and Overgaard J. OC-0278 Accelerated CH-RT with/without nimorazole for p16- HNSCC: the randomized DAHANCA 29-EORTC 1219 trial. *Radiother. Oncol.* **161**, S187–S188 (2021).
51. Bredell MG, Ernst J, El-Kochairi I, Dahlem Y, Ikenberg K, and Schumann DM. Current relevance of hypoxia in head and neck cancer. *Oncotarget* **7**, 50781–50804 (2016).
52. Janssen HL, Haustermans KM, Sprong D, Blommestijn G, Hofland I, Hoebers FJ, Blijweert E, Raleigh JA, Semenza GL, Varia MA, et al. HIF-1alpha;, pimonidazole, and iododeoxyuridine to estimate hypoxia and perfusion in human head-and-neck

tumors. *International Journal of Radiation Oncology*Biology*Physics* **54**, 1537–1549 (2002).

53. Bayer C and Vaupel P. Acute versus chronic hypoxia in tumors: Controversial data concerning time frames and biological consequences. *Strahlentherapie und Onkologie* **188**, 616–627 (2012).
54. Horsman MR and Overgaard J. The impact of hypoxia and its modification of the outcome of radiotherapy. *Journal of Radiation Research* **57**, i90–i98 (2016).
55. Zschaeck S, Löck S, Hofheinz F, Zips D, Saksø Mortensen L, Zöphel K, Troost EG, Boeke S, Saksø M, Mönnich D, et al. Individual patient data meta-analysis of FMISO and FAZA hypoxia PET scans from head and neck cancer patients undergoing definitive radio-chemotherapy. *Radiother. Oncol.* **149**, 189–196 (2020).
56. Löck S, Perrin R, Seidlitz A, Bandurska-Luque A, Zschaeck S, Zöphel K, Krause M, Steinbach J, Kotzerke J, Zips D, et al. Residual tumour hypoxia in head-and-neck cancer patients undergoing primary radiochemotherapy, final results of a prospective trial on repeat FMISO-PET imaging. *Radiotherapy and Oncology* **124**, 533–540 (2017).
57. Busk M, Horsman MR, and Overgaard J. Resolution in PET hypoxia imaging: Voxel size matters. *Acta Oncologica* **47**, 1201–1210 (2008).
58. Kaanders. Pimonidazole Binding and Tumor Vascularity and Predict for Treatment and Outcome in and Head and Neck and Cancer. *Cancer Res.* **62**, 7066–74 (2002).
59. Koi L, Bitto V, Weise C, Möbius L, Linge A, Löck S, Yaromina A, Besso MJ, Valentini C, Pfeifer M, et al. Prognostic biomarkers for the response to the radiosensitizer nimorazole combined with RCTx: a pre-clinical trial in HNSCC xenografts. *Journal of Translational Medicine* **21**. (2023).
60. Baumann M, Krause M, Overgaard J, Debus J, Bentzen SM, Daartz J, Richter C, Zips D, and Bortfeld T. Radiation oncology in the era of precision medicine. *Nat Rev Cancer* **16**, 234–249 (2016).
61. Clarke R, Ressom HW, Wang A, Xuan J, Liu MC, Gehan EA, and Wang Y. The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. *Nature Reviews Cancer* **8**, 37–49 (2008).
62. Wang Y, Miller DJ, and Clarke R. Approaches to working in high-dimensional data spaces: gene expression microarrays. *British Journal of Cancer* **98**, 1023–1028 (2008).

63. Yaqoob A, Musheer Aziz R, and verma NK. Applications and Techniques of Machine Learning in Cancer Classification: A Systematic Review. *Human-Centric Intelligent Systems*. (2023).
64. Nilsson R, Peña JM, Björkegren J, and Tegnër J. Consistent Feature Selection for Pattern Recognition in Polynomial Time. *Journal of Machine Learning Research* **8**, 589–612 (2007).
65. Saeys Y, Inza I, and Larrañaga P. A review of feature selection techniques in bioinformatics. *Bioinformatics* **23**, 2507–2517 (2007).
66. Guyon I and Elisseeff A. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research* 1157–1182 (2003).
67. Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, and Smyth GK. limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Res.* **43**, e47–e47 (2015).
68. Love MI, Huber W, and Anders S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology* **15**. (2014).
69. Chen X, Wang M, and Zhang H. The use of classification trees for bioinformatics. *WIREs Data Mining and Knowledge Discovery* **1**, 55–63 (2011).
70. Breiman L. Random Forests. *Machine Learning* **45**, 5–32 (2001).
71. Decision Tree. method compute_feature_importances (line 1268). scikit learn. URL: https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/tree/_tree.pyx (visited on 02/14/2024).
72. Chen X and Ishwaran H. Random forests for genomic data analysis. *Genomics* **99**, 323–329 (2012).
73. Nicodemus KK and Malley JD. Predictor correlation impacts machine learning algorithms: implications for genomic studies. *Bioinformatics* **25**, 1884–1890 (2009).
74. Strobl C, Boulesteix AL, Zeileis A, and Hothorn T. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics* **8**. (2007).
75. Genuer R, Poggi JM, and Tuleau-Malot C. Variable selection using random forests. *Pattern Recognition Letters* **31**, 2225–2236 (2010).
76. Kursa MB and Rudnicki WR. Feature Selection with the Boruta Package. *Journal of Statistical Software* **36**. (2010).

77. Degenhardt F, Seifert S, and Szymczak S. Evaluation of variable selection methods for random forests and omics data sets. *Briefings in Bioinformatics* **20**, 492–503 (2017).
78. Frades I and Matthiesen R. Overview on Techniques in Cluster Analysis. In: *Bioinformatics Methods in Clinical Research*. Methods in Molecular Biology. (2009):81–107. DOI: 10.1007/978-1-60327-194-3_5.
79. Chari T and Pachter L. The specious art of single-cell genomics. *PLOS Computational Biology* **19**, e1011288 (2023). Ed. by Papin JA.
80. Bhaskar H, Hoyle DC, and Singh S. Machine learning in bioinformatics: A brief survey and recommendations for practitioners. *Computers in Biology and Medicine* **36**, 1104–1125 (2006).
81. Larsen KG and Nelson J. Optimality of the Johnson-Lindenstrauss lemma. (2016). DOI: https://doi.org/10.48550/arXiv.1609.02094.
82. Yang L. Distance-preserving dimensionality reduction. *WIREs Data Mining and Knowledge Discovery* **1**, 369–380 (2011).
83. Jolliffe IT and Cadima J. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **374**, 20150202 (2016).
84. Abegaz F, Chaichoompu K, Génin E, Fardo DW, König IR, Mahachie John JM, and Van Steen K. Principals about principal components in statistical genetics. *Briefings in Bioinformatics* **20**, 2200–2216 (2018).
85. Elhaik E. Principal Component Analyses (PCA)-based findings in population genetic studies are highly biased and must be reevaluated. *Scientific Reports* **12**. (2022).
86. Thomas SA, Race AM, Steven RT, Gilmore IS, and Bunch J. Dimensionality reduction of mass spectrometry imaging data using autoencoders. In: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, (2016). DOI: 10.1109/ssci.2016.7849863.
87. Matsuda K and Aoyagi S. Sparse autoencoder–based feature extraction from TOF–SIMS image data of human skin structures. *Analytical and Bioanalytical Chemistry* **414**, 1177–1186 (2022).
88. McInnes L, Healy J, and Melville J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. (2018). DOI: 10.48550/ARXIV.1802.03426.

89. Maaten L van der and Hinton G. Visualizing Data using t-SNE. *Journal of Machine Learning Research* **9**, 2579–2605 (2008).
90. Nelles O. Neural Networks. In: *Nonlinear System Identification*. Springer International Publishing, (2020):279–345. DOI: 10.1007/978-3-030-47439-3_11.
91. Gokcesu K and Gokcesu H. Generalized Huber Loss for Robust Learning and its Efficient Minimization for a Robust Statistics. (2021). DOI: 10.48550/ARXIV.2108.12627.
92. Saleh H. The Machine Learning Workshop. Get ready to develop your own high-performance machine learning algorithms with scikit-learn. 2. Birmingham: Packt Publishing Limited, (2020):164–5. 1286 pp.
93. Albawi S, Mohammed TA, and Al-Zawi S. Understanding of a convolutional neural network. In: *2017 International Conference on Engineering and Technology (ICET)*. IEEE, (2017). DOI: 10.1109/icengtechnol.2017.8308186.
94. O'Shea K and Nash R. An Introduction to Convolutional Neural Networks. (2015). DOI: 10.48550/ARXIV.1511.08458.
95. Li Z, Liu F, Yang W, Peng S, and Zhou J. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems* **33**, 6999–7019 (2022).
96. Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, Santamaría J, Fadhel MA, Al-Amidie M, and Farhan L. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data* **8**. (2021).
97. Huang G, Liu Z, Maaten L van der, and Weinberger KQ. Densely Connected Convolutional Networks. (2016). DOI: 10.48550/ARXIV.1608.06993.
98. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, and Rabinovich A. Going Deeper with Convolutions. (2014). DOI: 10.48550/ARXIV.1409.4842.
99. Lin M, Chen Q, and Yan S. Network In Network. (2013). DOI: 10.48550/ARXIV.1312.4400.
100. Huh M, Agrawal P, and Efros AA. What makes ImageNet good for transfer learning? (2016). DOI: 10.48550/ARXIV.1608.08614.
101. Lundberg S and Lee SI. A Unified Approach to Interpreting Model Predictions. (2017). DOI: 10.48550/ARXIV.1705.07874.

102. Plaut E. From Principal Subspaces to Principal Components with Linear Autoencoders. (2018). DOI: 10.48550/ARXIV.1804.10253.
103. Bank D, Koenigstein N, and Giryes R. Autoencoders. (2020). DOI: 10.48550/ARXIV.2003.05991.
104. Cavallari GB, Ribeiro LSF, and Ponti MA. Unsupervised representation learning using convolutional and stacked auto-encoders: a domain and cross-domain feature space analysis. (2018). DOI: 10.48550/ARXIV.1811.00473.
105. Xiao S, Wang S, and Guo W. SGAE: Stacked Graph Autoencoder for Deep Clustering. *IEEE Transactions on Big Data* **9**, 254–266 (2023).
106. Klein S, Staring M, Murphy K, Viergever M, and Pluim J. elastix: A Toolbox for Intensity-Based Medical Image Registration. *IEEE Transactions on Medical Imaging* **29**, 196–205 (2010).
107. Lester H and Arridge SR. A survey of hierarchical non-linear medical image registration. *Pattern Recognition* **32**, 129–149 (1999).
108. Bairoch A. Cellosaurus - a knowledge resource on cell lines. CALIPHO group at the SIB - Swiss Institute of Bioinformatics. URL: https://web.expasy.org/cellosaurus/ (visited on 02/14/2023).
109. Takahashi K, Kanazawa H, Akiyama Y, Tazaki S, Takahara M, Muto T, Tanzawa H, and Sato Ki. Establishment and characterization of a cell line (SAS) from poorly differentiated human squamous cell carcinoma of the tongue. *J. Jpn. Stomatol. Soc.* **38**, 20–28 (1989).
110. Perez-Riverol Y, Bai J, Bandla C, García-Seisdedos D, Hewapathirana S, Kamatchinathan S, Kundu DJ, Prakash A, Frericks-Zipper A, Eisenacher M, et al. The PRIDE database resources in 2022: a hub for mass spectrometry-based proteomics evidences. *Nucleic Acids Research* **50**, D543–D552 (2021).
111. Bitto V, Hönscheid P, Besso MJ, Sperling C, Kurth I, Baumann M, and Brors B. Easing accessibility to mass spectrometry imaging using convolutional autoencoders for deriving hypoxia-associated peptide candidates from tumors. (2024). DOI: 10.21203/rs.3.rs-3755587/v1.
112. Map the whole transcriptome within the tissue context. 10x Genomics. URL: https://www.10xgenomics.com/products/spatial-gene-expression (visited on 12/04/2023).

113. Bankhead P, Loughrey MB, Fernández JA, Dombrowski Y, McArt DG, Dunne PD, McQuaid S, Gray RT, Murray LJ, Coleman HG, et al. QuPath: Open source software for digital pathology image analysis. *Scientific Reports* **7**. (2017).
114. Bradski G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. (2000).
115. Ntatsis K, Dekker N, Valk V van der, Birdsong T, Zukić D, Klein S, Staring M, and McCormick M. itk-elastix: Medical image registration in Python. In: *Proceedings of the 22nd Python in Science Conference*. Ed. by Agarwal M, Calloway C, and Niederhut D. (2023):101–5. DOI: 10.25080/gerudo-f2bc6f59-00d.
116. Shamonin DP, Bron EE, Lelieveldt BPF, Smits M, Klein S, and Staring M. Fast parallel image registration on CPU and GPU for diagnostic classification of Alzheimer's disease. *Frontiers in Neuroinformatics* **7**, 1–15 (2014).
117. Bemis KD, Harry A, Eberlin LS, Ferreira C, Ven SM van de, Mallick P, Stolowitz M, and Vitek O. Cardinal: an R package for statistical analysis of mass spectrometry-based imaging experiments: Fig. 1. *Bioinformatics* **31**, 2418–2420 (2015).
118. Space Ranger Support. 10x Genomics. URL: https://www.10xgenomics.com/support/software/space-ranger (visited on 02/14/2023).
119. Hafemeister C and Satija R. Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression. *Genome Biology* **20**. (2019).
120. Wolf FA, Angerer P, and Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biology* **19**. (2018).
121. Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, Niculae V, Prettenhofer P, Gramfort A, Grobler J, et al. API design for machine learning software: experiences from the scikit-learn project. In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. (2013):108–22.
122. McKinney W. Data Structures for Statistical Computing in Python. In: *Proceedings of the 9th Python in Science Conference*. Ed. by Walt S van der and Millman J. (2010):56–61. DOI: 10.25080/Majora-92bf1922-00a.
123. Harris CR, Millman KJ, Walt SJ van der, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, et al. Array programming with NumPy. *Nature* **585**, 357–362 (2020).

124. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* **17**, 261–272 (2020).
125. Virshup I, Rybakov S, Theis FJ, Angerer P, and Wolf FA. anndata: Annotated data. (2021).
126. Homola D. boruta_py. URL: https://github.com/scikit-learn-contrib/boruta_py (visited on 02/14/2024).
127. Bitto V. GitHub - automsi package. URL: https://github.com/DKFZ-ABI/automsi (visited on 02/23/2024).
128. Tyanova S, Temu T, and Cox J. The MaxQuant computational platform for mass spectrometry-based shotgun proteomics. *Nature Protocols* **11**, 2301–2319 (2016).
129. Montesinos López OA, Montesinos López A, and Crossa J. Random Forest for Genomic Prediction. In: *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Springer International Publishing, (2022):633–81. DOI: 10.1007/978-3-030-89010-0_15.
130. Mascini NE and Heeren RM. Protein identification in mass-spectrometry imaging. In: vol. 40. Elsevier BV, (2012):28–37. DOI: 10.1016/j.trac.2012.06.008.
131. Veselkov K, Sleeman J, Claude E, Vissers JPC, Galea D, Mroz A, Laponogov I, Towers M, Tonge R, Mirnezami R, et al. BASIS: High-performance bioinformatics platform for processing of large-scale mass spectrometry imaging data in chemically augmented histology. *Scientific Reports* **8**. (2018).
132. Brenton AG and Godfrey AR. Accurate mass measurement: Terminology and treatment of data. *Journal of the American Society for Mass Spectrometry* **21**, 1821–1835 (2010).
133. Chen Z, Han F, Du Y, Shi H, and Zhou W. Hypoxic microenvironment in cancer: molecular mechanisms and therapeutic interventions. *Signal Transduction and Targeted Therapy* **8**. (2023).
134. Cuthbertson CR, Arabzada Z, Bankhead A, Kyani A, and Neamati N. A Review of Small-Molecule Inhibitors of One-Carbon Enzymes: SHMT2 and MTHFD2 in the Spotlight. *ACS Pharmacology & Translational Science* **4**, 624–646 (2021).
135. Raulf N, Lucarelli P, Thavaraj S, Brown S, Vicencio J, Sauter T, and Tavassoli M. Annexin A1 regulates EGFR activity and alters EGFR-containing tumour-derived exosomes in head and neck cancers. *European Journal of Cancer* **102**, 52–68 (2018).

136. Higashimura Y, Nakajima Y, Yamaji R, Harada N, Shibasaki F, Nakano Y, and Inui H. Up-regulation of glyceraldehyde-3-phosphate dehydrogenase gene expression by HIF-1 activity depending on Sp1 in hypoxic breast cancer cells. *Archives of Biochemistry and Biophysics* **509**, 1–8 (2011).
137. Space Ranger Spatial Outputs. 10x Genomics. URL: https://www.10xgenomics.com/support/software/space-ranger/latest/analysis/outputs/spatial-outputs (visited on 12/21/2023).
138. Choudhary S and Satija R. Comparison and evaluation of statistical error models for scRNA-seq. *Genome Biology* **23**. (2022).
139. Mase M, Owen AB, and Seiler B. Explaining black box decisions by Shapley cohort refinement. (2019). DOI: 10.48550/ARXIV.1911.00467.
140. Aas K, Jullum M, and Løland A. Explaining individual predictions when features are dependent: More accurate approximations to Shapley values. *Artificial Intelligence* **298**, 103502 (2021).
141. Gardner W, Winkler DA, Cutts SM, Torney SA, Pietersz GA, Muir BW, and Pigram PJ. Two-Dimensional and Three-Dimensional Time-of-Flight Secondary Ion Mass Spectrometry Image Feature Extraction Using a Spatially Aware Convolutional Autoencoder. *Analytical Chemistry* **94**, 7804–7813 (2022).
142. Li Y, Gan Z, Zhou X, and Chen Z. Accurate classification of Listeria species by MALDI-TOF mass spectrometry incorporating denoising autoencoder and machine learning. *Journal of Microbiological Methods* **192**, 106378 (2022).
143. Hu H and Laskin J. Emerging Computational Methods in Mass Spectrometry Imaging. *Advanced Science* **9**. (2022).
144. Behrmann J, Etmann C, Boskamp T, Casadonte R, Kriegsmann J, and Maa P. Deep learning for tumor classification in imaging mass spectrometry. *Bioinformatics* **34**, 1215–1223 (2017).
145. Zanjani FG, Panteli A, Zinger S, Sommen Fvd, Tan T, Balluff B, Vos DRN, Ellis SR, Heeren RMA, Lucas M, et al. Cancer Detection in Mass Spectrometry Imaging Data by Recurrent Neural Networks. In: *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE, (2019). DOI: 10.1109/isbi.2019.8759571.
146. Williams CG, Lee HJ, Asatsuma T, Vento-Tormo R, and Haque A. An introduction to spatial transcriptomics for biomedical research. *Genome Medicine* **14**. (2022).

147. Li Y, Stanojevic S, and Garmire LX. Emerging artificial intelligence applications in Spatial Transcriptomics analysis. *Computational and Structural Biotechnology Journal* **20**, 2895–2908 (2022).
148. Xu H, Fu H, Long Y, Ang KS, Sethi R, Chong K, Li M, Uddamvathanak R, Lee HK, Ling J, et al. Unsupervised spatially embedded deep representation of spatial transcriptomics. *Genome Medicine* **16**. (2024).
149. Dong K and Zhang S. Deciphering spatial domains from spatially resolved transcriptomics with an adaptive graph attention auto-encoder. *Nature Communications* **13**. (2022).
150. Kassuhn W, Klein O, Darb-Esfahani S, Lammert H, Handzik S, Taube ET, Schmitt WD, Keunecke C, Horst D, Dreher F, et al. Classification of Molecular Subtypes of High-Grade Serous Ovarian Cancer by MALDI-Imaging. *Cancers* **13**, 1512 (2021).
151. Hoffmann F, Umbreit C, Krüger T, Pelzel D, Ernst G, Kniemeyer O, Guntinas-Lichius O, Berndt A, and Eggeling F. Identification of Proteomic Markers in Head and Neck Cancer Using MALDI–MS Imaging, LC–MS/MS, and Immunohistochemistry. *PROTEOMICS – Clinical Applications* **13**, 1700173 (2018).
152. Djidja MC, Chang J, Hadjiprocopis A, Schmich F, Sinclair J, Mršnik M, Schoof EM, Barker HE, Linding R, Jørgensen C, et al. Identification of Hypoxia-Regulated Proteins Using MALDI-Mass Spectrometry Imaging Combined with Quantitative Proteomics. *Journal of Proteome Research* **13**, 2297–2313 (2014).
153. Mascini NE, Cheng M, Jiang L, Rizwan A, Podmore H, Bhandari DR, Römpp A, Glunde K, and Heeren RM. Mass Spectrometry Imaging of the Hypoxia Marker Pimonidazole in a Breast Tumor Model. *Analytical Chemistry* **88**, 3107–3114 (2016).
154. Morrison LE, Lefever MR, Lewis HN, Kapadia MJ, and Bauer DR. Conventional histological and cytological staining with simultaneous immunohistochemistry enabled by invisible chromogens. *Laboratory Investigation* **102**, 545–553 (2021).
155. Patterson NH, Tuck M, Plas RV de, and Caprioli RM. Advanced Registration and Analysis of MALDI Imaging Mass Spectrometry Measurements through Autofluorescence Microscopy. *Analytical Chemistry* **90**, 12395–12403 (2018).
156. Race AM, Sutton D, Hamm G, Maglennon G, Morton JP, Strittmatter N, Campbell A, Sansom OJ, Wang Y, Barry ST, et al. Deep Learning-Based Annotation Transfer between Molecular Imaging Modalities: An Automated Workflow for Multimodal Data Integration. *Analytical Chemistry* **93**, 3061–3071 (2021).

157. Cordes J, Enzlein T, Marsching C, Hinze M, Engelhardt S, Hopf C, and Wolf I. M2aia—Interactive, fast, and memory-efficient analysis of 2D and 3D multi-modal mass spectrometry imaging data. *GigaScience* **10**. (2021).
158. Nolden M, Zelzer S, Seitel A, Wald D, Müller M, Franz AM, Maleike D, Fangerau M, Baumhauer M, Maier-Hein L, et al. The Medical Imaging Interaction Toolkit: challenges and advances: 10 years of open-source development. *International Journal of Computer Assisted Radiology and Surgery* **8**, 607–620 (2013).
159. Higgins I, Matthey L, Pal A, Burgess C, Glorot X, Botvinick M, Mohamed S, and Lerchner A. Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. (2017).
160. Buccitelli C and Selbach M. mRNAs, proteins and the emerging principles of gene expression control. *Nature Reviews Genetics* **21**, 630–644 (2020).
161. Zhou R, Yang G, Zhang Y, and Wang Y. Spatial transcriptomics in development and disease. *Molecular Biomedicine* **4**. (2023).
162. Dario Bressan Giorgia Battistoni GJH. The dawn of spatial omics. *Science* **381**. (2023).
163. elastix. SimilarityTransforms in elastix. URL: https://elastix.lumc.nl/doxygen/classelastix_1_1SimilarityTransformElastix.html (visited on 02/22/2024).
164. elastix. SimilarityTransforms in ITK. URL: https://itk.org/Doxygen/html/classitk_1_1Similarity2DTransform.html (visited on 02/22/2024).
165. Sundstrom A, Grabocka E, Bar-Sagi D, and Mishra B. Histological Image Processing Features Induce a Quantitative Characterization of Chronic Tumor Hypoxia. *PLOS ONE* **11**, e0153623 (2016). Ed. by Chammas R.
166. Manescu P, Geradts J, and Fernandez-Reyes D. Deep learning-based detection of morphological features associated with hypoxia in H&E breast cancer whole slide images. (2023). DOI: 10.48550/ARXIV.2311.12601.
167. Echle A, Rindtorff NT, Brinker TJ, Luedde T, Pearson AT, and Kather JN. Deep learning in cancer pathology: a new generation of clinical biomarkers. *British Journal of Cancer* **124**, 686–696 (2020).
168. Kather JN, Pearson AT, Halama N, Jäger D, Krause J, Loosen SH, Marx A, Boor P, Tacke F, Neumann UP, et al. Deep learning can predict microsatellite

instability directly from histology in gastrointestinal cancer. *Nature Medicine* **25**, 1054–1056 (2019).

169. Monjo T, Koido M, Nagasawa S, Suzuki Y, and Kamatani Y. Efficient prediction of a spatial transcriptomics profile better characterizes breast cancer tissue sections without costly experimentation. *Scientific Reports* **12**. (2022).
170. He B, Bergenstråhle L, Stenbeck L, Abid A, Andersson A, Borg Å, Maaskola J, Lundeberg J, and Zou J. Integrating spatial gene expression and breast tumour morphology via deep learning. *Nature Biomedical Engineering* **4**, 827–834 (2020).
171. Schneider L, Laiouar-Pedari S, Kuntz S, Krieghoff-Henning E, Hekler A, Kather JN, Gaiser T, Fröhling S, and Brinker TJ. Integration of deep learning-based image analysis and genomic data in cancer pathology: A systematic review. *European Journal of Cancer* **160**, 80–91 (2022).
172. Bergenstråhle L, He B, Bergenstråhle J, Abalo X, Mirzazadeh R, Thrane K, Ji AL, Andersson A, Larsson L, Stakenborg N, et al. Super-resolved spatial transcriptomics by deep data fusion. *Nature Biotechnology* **40**, 476–479 (2021).
173. Plas RV de, Yang J, Spraggins J, and Caprioli RM. Image fusion of mass spectrometry and microscopy: a multimodality paradigm for molecular tissue mapping. *Nature Methods* **12**, 366–372 (2015).
174. Feldner-Busztin D, Firbas Nisantzis P, Edmunds SJ, Boza G, Racimo F, Gopalakrishnan S, Limborg MT, Lahti L, and Polavieja GG de. Dealing with dimensionality: the application of machine learning to multi-omics data. *Bioinformatics* **39**. (2023). Ed. by Wren J.
175. Xu J, Wu P, Chen Y, Meng Q, Dawood H, and Dawood H. A hierarchical integration deep flexible neural forest framework for cancer subtype classification by integrating multi-omics data. *BMC Bioinformatics* **20**. (2019).
176. Picard M, Scott-Boyer MP, Bodein A, Périn O, and Droit A. Integration strategies of multi-omics data for machine learning analysis. *Computational and Structural Biotechnology Journal* **19**, 3735–3746 (2021).
177. Shurrab S and Duwairi R. Self-supervised learning methods and applications in medical imaging analysis: a survey. *PeerJ Computer Science* **8**, e1045 (2022).
178. Zhou W, Zhao C, Liu A, Zhang X, Cao X, Ding Z, Sha Q, Shen H, and Deng HW. CLCLSA: Cross-omics Linked embedding with Contrastive Learning and Self

Attention for multi-omics integration with incomplete multi-omics data. (2023). DOI: 10.21203/rs.3.rs-2768563/v1.

179. Lee B, Zhang S, Poleksic A, and Xie L. Heterogeneous Multi-Layered Network Model for Omics Data Integration and Analysis. *Frontiers in Genetics* **10**. (2020).
180. Rajadhyaksha N and Chitkara A. Graph Contrastive Learning for Multi-omics Data. (2023). DOI: 10.48550/ARXIV.2301.02242.
181. National Cancer Institute Dictionary of Cancer Terms. National Cancer Institute. URL: https://www.cancer.gov/publications/dictionaries/cancer-terms/ (visited on 02/14/2023).
182. Deininger SO, Cornett DS, Paape R, Becker M, Pineau C, Rauser S, Walch A, and Wolski E. Normalization in MALDI-TOF imaging datasets of proteins: practical considerations. *Analytical and Bioanalytical Chemistry* **401**, 167–181 (2011).