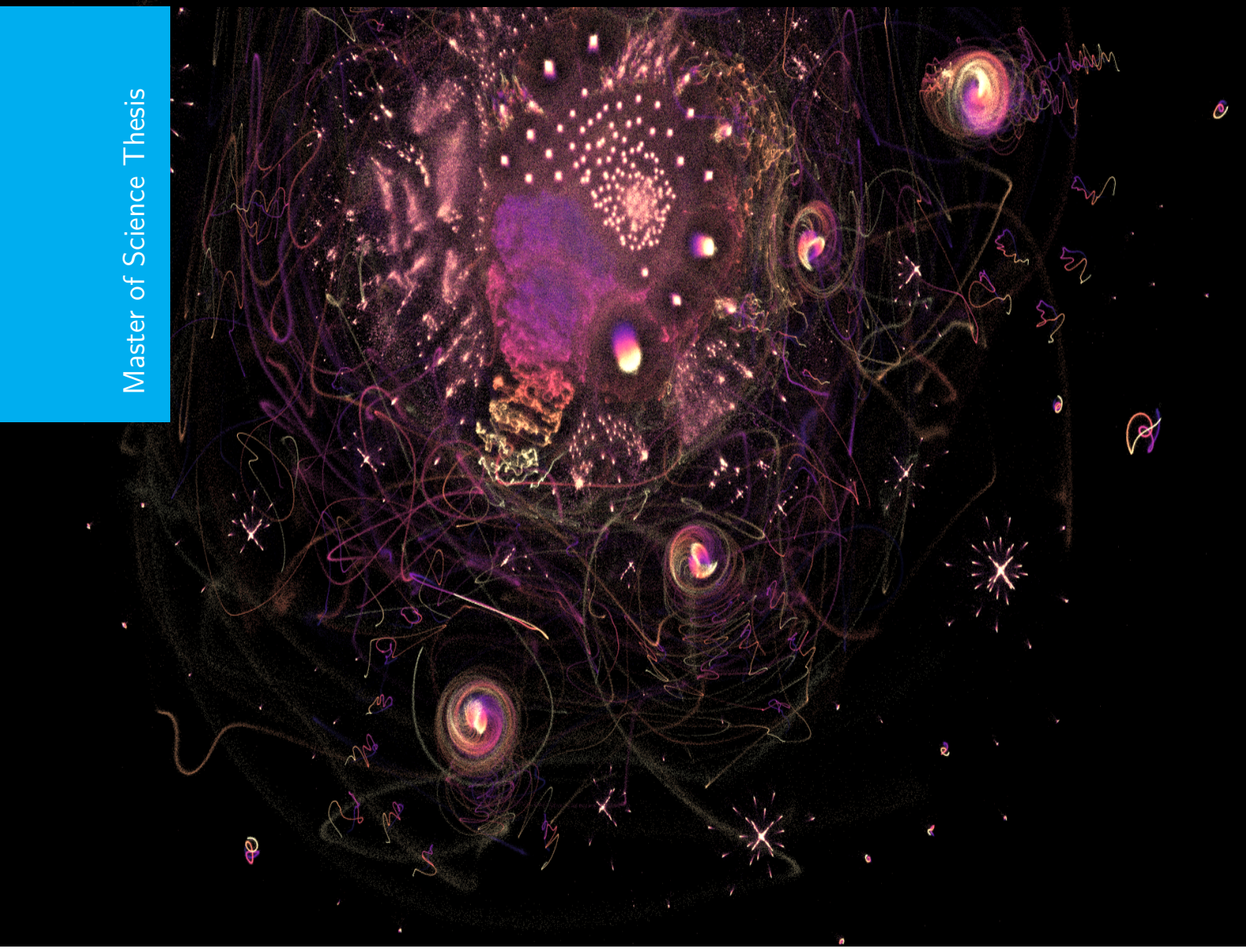
Data-driven Parameter Optimization and Spatial Awareness for Uniform Manifold Approximation and Projection (UMAP) of IMS data sets.

A step towards parameter-free dimensionality reduction

S.T. Jansen

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Data-driven Parameter Optimization and Spatial Awareness for Uniform Manifold Approximation and Projection (UMAP) of IMS data sets.

A step towards parameter-free dimensionality reduction

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

S.T. Jansen

September 30, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Cover image: The first 8 million integers, represented as binary vectors indicating their prime factors, and laid out using UMAP. By: John Williamson. URL: https://johnhw.github.io/umap_primes/index.md.html

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Data-driven Parameter Optimization and Spatial Awareness for Uniform Manifold Approximation and Projection (UMAP) of IMS data sets.

by

S.T. Jansen

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: September 30, 2021

Supervisor(s):

dr.ir. R. Van de Plas

dr. L.G. Migas

Reader(s):

dr. M.W.E.M. Alfeld

Abstract

Imaging Mass Spectrometry (IMS) is a powerful technique capable of extracting unlabeled spatial and chemical information from a biological tissue sample. Ever-increasing technological advancements have resulted in rapid growth of IMS data set sizes, scaling quadratically with the increasingly refined spatial resolution of its ion images. Dimensionality reduction techniques aim to make large data sets practically approachable by reducing the number of dimensions in the data while retaining as much information as possible. Nonlinear Dimensionality Reduction (NLDR) methods attempt to uncover an underlying nonlinear manifold or structure in the data by constructing a Low-Dimensional (LD) feature space with variables that are nonlinear combinations of the original features (i.e. mass-to-charge ratio (m/z) bins in IMS measurements). The t-Distributed Stochastic Neighbor Embedding (t-SNE) has been a common approach in NLDR methods, using force-directed graphs. Uniform Manifold Approximation and Projection (UMAP) works in a similar manner, but leans on a more versatile mathematical foundation in topology, is generally faster, and the method tends to scale better than t-SNE.

In this thesis, we introduce UMAPLUS, an extension of UMAP that takes not only spectral, but also spatial information into account. Our experiments show that UMAPLUS is capable of extracting the structure of a synthetic IMS data set better than UMAP. Besides UMAP's standard spectral and random initializations, we introduce a new spatial initialization that provides more intuitive insight into the LD embedding relative to the spatial image domain. Standard UMAP entails several parameters that influence consistency, reliability, and quality of its LD embedding. Thus far, the influence of these parameters on IMS data sets has been barely investigated, and previous studies have consistently applied the default settings of UMAP. In this thesis, a Data-Driven UMAP (DD-UMAP) is constructed, which includes optimization of UMAP parameters in a data-driven manner. Several distance metrics are investigated, with cosine similarity providing the most robust results. A naive 1-D optimization procedure is compared to a multivariate Bayesian optimization approach, capable of optimizing multiple parameters simultaneously. Using an evaluation function partly based on the cost function of UMAP and the addition of spatial information, DD-UMAP is able to estimate and utilize an optimized input parameter set for UMAP in an unsupervised manner and on unlabeled IMS data sets. This is demonstrated on generated synthetic data, and two real-world IMS data sets; a full mouse pup and a murine kidney.

Table of Contents

Acknowledgements

With this thesis, I conclude my MSc Systems & Control at Delft University of Technology, and simultaneously my journey of many years in Delft. Last year I learned a lot from a topic that was quite unknown to me and would be unnoticed if it was not for my supervisors Raf Van de Plas and Lukasz Migas. Both have been a major virtual support throughout my thesis, through critical comments, drawing attention of not drinking too much water during presentations, and guiding me into the right direction of tackling a project of this scope. Thank you for being my mentors. It has been a year with many ups and down, but in my opinion the most challenging and interesting part would be the commitment to a single long-term topic.

Secondly, I would like to show appreciation to Katerina Djambazova, Elizabeth Neumann, Jeffrey Spraggins and Richard Caprioli for providing the real-world IMS data sets of the mouse pup and the murine kidney which were acquired at Vanderbilt University, and are used throughout this thesis.

Finally, I would like to thank my friends and family for helping me through this weird time where the whole world was upside-down. In a time when lots of activities were restricted, from having a decent place to study to loosen up after a day of working, they helped me get my mind of any form of nonlinear dimensionality reduction algorithms. Last but definitely not least, I would like to thank Fieke for staying at my side through all fluctuations during this project. Please know I never took it for granted.

Delft, University of Technology S.T. Jansen
September 30, 2021

Chapter 1

Introduction

Technological advancements in general have led to experiments with ever-increasing data set size, with the idea that more information in the form of more measurements is linked to better insights. However, storing large amounts of data becomes eventually problematic, and even unworkable. Further developments on the computational power of workstations have relieved these concerns slightly. Nevertheless, due to the rapid expansion and growing need for better intuitive insights, extracting and retaining only the most important features from a data set may offer a solution. Dimensionality reduction methods seeks a way to reduce the number of features of the high-dimensional data, while trying to retain as much information of the original features as possible [1]. With the reduced data, better interpretability due to clearer visualizations also becomes more feasible.

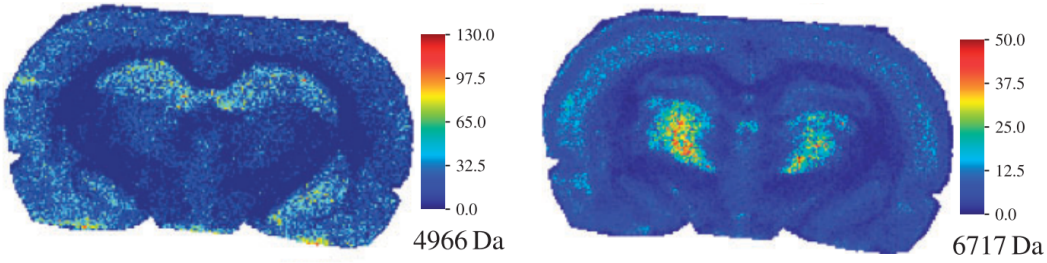


Figure 1-1: Two examples of an ion image, both taken at different m/z values, which are usually measured in Dalton (Da). The absolute intensity at a specific m/z bin is in arbitrary units [2].

Imaging Mass Spectrometry (IMS) is one such technology suffering from the incremental data set size. In IMS, a biological tissue is sampled by a predefined virtual two-dimensional grid defined on top of the surface, measuring its chemical content at each spatial location ((x, y) coordinates). Through multiple steps, which will be discussed in section 2-1, each measurement contains a mass spectrum quantifying said chemical content. The presence of the measured ions are characterized by a certain ion intensity value of a specific mass-to-charge ratio (m/z). An ion image visualizes the ion intensity distribution at a specific m/z, as

can be seen in Figure 1-1. IMS data sets can easily consist of $10^3 - 10^6$ pixels of each $10^4 - 10^6$ m/z bins, reaching in the GBs for a single tissue-wide measurement [3]. Increasing spatial resolution of newly developed IMS instruments cause the IMS data sets to scale quadratically in the case of 2-D experiments [4].

In terms of unsupervised data mining of IMS (in cases where biomarkers are not known a priori), a distinction between dimensionality reduction techniques can be made between linear or nonlinear decomposition methods. A popular linear technique broadly used in IMS is Principal Component Analysis (PCA), which aims to extract a new representation of a measurement set using only a reduced number of features from the original high-dimensional data, maximizing the variance in the Low-Dimensional (LD) space [5]. While this technique is sufficient when the data mixes in a locally linear fashion [3], Nonlinear Dimensionality Reduction (NLDR) techniques are superior in extracting an underlying nonlinear manifold (if present). We define local linearity over a sufficiently small interval such that a tangent line approximates the underlying nonlinear function.

For over 10 years, t-Distributed Stochastic Neighbor Embedding (t-SNE) has been intensively used throughout the literature as a basis for NLDR techniques [6]. t-SNE is built upon its predecessor Stochastic Neighbor Embedding (SNE) [7]. t-SNE uses a probabilistic approach, and its clear separated clusters have made it attractive for visualisation purposes. However, the algorithm has a computational complexity of $O(N^2)$ with N the number of data points, resulting in long runtimes for large data sets. This process has been sped up by several variations of t-SNE like Barnes-Hut SNE [8] and FIt-SNE [9], but one criticism has been that the foundation of t-SNE is not well understood [10]. Uniform Manifold Approximation and Projection (UMAP) aims to resolve this by introducing a method with a rather well founded mathematically theory in topology, which is generally faster and scales better than t-SNE [11].

UMAP introduces many parameters that influence the consistency, reliability and quality of the produced LD embedding of the data. More influential parameters are called hyperparameters. In the literature, the default values of these hyperparameters are primarily used without investigation of potential improvements to the LD embedding. Opt-SNE has been the pioneer in nonlinear data-driven parameter optimization, built upon t-SNE [12]. Utilizing its cost function (Kullback–Leibler divergence (KLD)), opt-SNE is able to calibrate the early exaggeration parameter which has been fixed in the original version of t-SNE. A control strategy for optimizing UMAP parameters is yet to be implemented, a main topic of this thesis that will lead to Data-Driven UMAP (DD-UMAP). Since the parameters are assumed to be dependent of each other, a multivariate Sequential Model Based Optimisation (SMBO) framework will most likely be the more promising approach than applying sequential 1-D optimization for the parameter search-space strategy. This, in combination with the unsupervised scoring metric of LD embeddings, will be the main challenges.

With the acquired mass spectra, not only the chemical content of each pixel is known, we also know the position of each individual pixel. Up until now, this spatial information is yet to be incorporated into NLDR techniques. Maximum Autocorrelation Factorization (MAF) is known to have maximum spatial autocorrelation as an objective function, however MAF is a linear decomposition technique and insufficient to extract a latent LD nonlinear manifold within High-Dimensional (HD) data (if present). This thesis will introduce UMAPLUS, which is an extension to UMAP and aims to also utilize the spatial structure (e.g. the

pixel coordinates) of an IMS data set. UMAPLUS introduces a new parameter $\lambda_{spatial}$ to the algorithm, which interpolates between the prioritization of applying forces on the LD embedding based upon either spectral or spatial information.

Research objectives The introduction of DD-UMAP and UMAPLUS will be the main extensions of UMAP, and the consecutive objective of this thesis can be summarized in the following statement:

> Developing an effective framework to automatically estimate the hyperparameters of Uniform Manifold Approximation and Projection (UMAP) for Imaging Mass Spectrometry (IMS) data sets with maximum usage of the available information, adaptable to different data sets, and therefore be able to extract the best possible LD embedding from the given sample in terms of its underlying (biological) structure.

In order to solve this problem, the following four sub-questions arise. These will be investigated in more detail to substantiate the main objective.

1. To what extent does the initialization phase have an effect on the produced embedding, and is there a better way of initializing than the standard procedure of Uniform Manifold Approximation and Projection (UMAP)?
2. How sensitive is UMAP to various parameter configurations? Can we classify whether a produced embedding is considered "better" than another, possibly with a combination of spectral and spatial information?
3. Is it possible to extend UMAP to incorporate spatial information during the optimization of the LD embedding, and can a better LD embedding be achieved using UMAPLUS? If so, what is the sensitivity of parameters `n_neighbors_spatial` and $\lambda_{spatial}$?
4. Is Data-Driven UMAP (DD-UMAP) able to find the optimal hyperparameters and thus produce the best possible embedding? What optimization method (1-D or Bayesian optimization approach) is more suitable to find the global optimum in the defined parameter search-space? And how does DD-UMAP compare to traditional grid search?

Thesis outline This thesis will start with defining the fundamentals of Imaging Mass Spectrometry (IMS). Chapter 2 will summarize the many steps taken in an IMS workflow, as well as the structure of IMS data sets. The fundamentals also include the explanation and motivation for Dimension Reduction (DR) techniques. In chapter 3, the methods of this thesis will be discussed. First, we will go through the entire UMAP algorithm and the assessment of the LD embedding applied for further optimization. Then, UMAPLUS and DD-UMAP are introduced. A scoring metric is constructed, able to evaluate the produced LD embedding. A complete overview of the parameters within UMAP will be constructed, along with possible optimization methods to determine the optimal parameter configuration. Each of the 4 sub-questions and the effect of UMAPLUS and DD-UMAP on synthetic and real-world IMS data sets will be investigated in their own separate experiments before the presentation of the results, discussion and recommendations for future work.

Chapter 2

Fundamentals

2-1 Imaging Mass Spectrometry (IMS)

Imaging Mass Spectrometry (IMS) is a label free technique that makes it possible to analyze the spatial distribution of the analytes as well as registering the molecular content of a given tissue sample. IMS is a complex process, and multiple methods exist in obtaining the chemical information from the tissue surface. An IMS data set is build up by the acquired mass spectrum for every sampled pixel in the image. This will be more explained in more detail with visual assistance in subsection 2-1-2. The mass-to-charge ratio (m/z) is measured by the IMS instrument, which is equal to the mass of an ion divided by its charge. A single ion image typically consist of 100k−500k pixels, each containing a spectrum with thousands of m/z bins. When selecting a specific m/z bin over the whole image, an ion image can be constructed as shown in Figure 1-1.

2-1-1 General IMS procedure

The main goal of IMS is to translate the analytical power of mass spectrometry to chemical images illustrating the distribution of the molecules spatially, and display the intensity at the corresponding m/z bins [13]. The minimization of the spatial resolution of the ion image is a requirement that has been tried to improve throughout the years. The spatial resolution is defined by a predefined 2-D array that is spread out over the tissue sample. At those discrete spots, ionization takes place. Combining both molecular and spatial information, IMS is capable of visualizing a wide range of molecules including lipids, peptides and proteins [14].

This section aims to explain the essence of IMS, why it is useful today, and go over most of the steps taken in a typical IMS workflow (see Figure 2-1). First, several techniques exist in the IMS field which obtain the mass spectra. These techniques will be discussed in subsubsection 2-1-1-2. The form of the obtained IMS data set will shortly be described in subsection 2-1-2. The raw data may contain biases or imperfections, which can partly be corrected by preprocessing, as will be discussed in subsubsection 2-1-1-4.

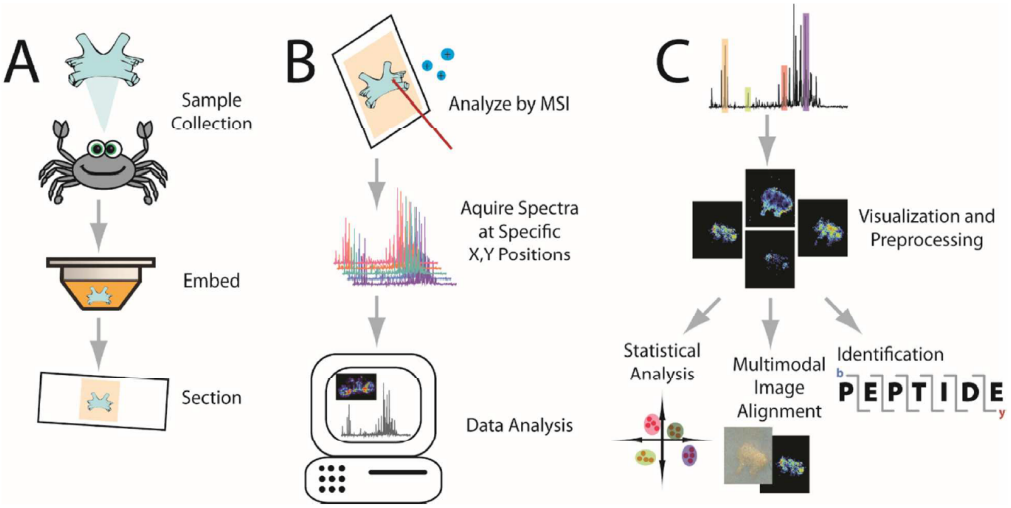


Figure 2-1: Visualisation of the workflow of IMS analysis which consist of three major steps. **(A)** Sample preparation; After collection from the animal, the sample is embedded in a supporting medium for sectioning onto slides. **(B)** Sample analysis; through ionization the analytes are extracted from the tissue surface by IMS, acquiring a spectrum at each (x, y) grid point on the tissue. Sophisticated software tools are used to process and visualize the data. **(C)** Data processing; After preprocessing the data, the distribution of selected ions can be visualized. From there on, identification of the m/z values, statistical analysis between different images, or image co-registration with other image modalities can be executed [1].

2-1-1-1 Sample preparation

Prior to collecting data using IMS, the tissue sample has to be prepared in the correct way for the corresponding instrument. Preparation of the biological tissue sample is necessary in order to reduce degradation and diffusion of the ions across the tissue [1]. Swales et al. stresses that in order to strive for reproducible sampling results, collection protocols exist to maintain tissue integrity and minimize any endogenous or exogenous compound degredation within the tissue [14]. Protocols for structurally preparing tissue samples have been described for MALDI and SIMS [15].

This section will go over different methods to collect tissue samples, discuss the ideal storage conditions to prevent long-term degradation, and explain the importance of cryo-sectioning the tissue sample prior to data acquisition by IMS.

Collection/ storage Following necroscopy, the desired samples can be snap frozen or fixed in formalin. By flash-freezing, the sample will typically be submerged in liquid nitrogen or dry-iced chilled isopentane [14]. Depending on the size of the sample, the user has the freedom to choose the appropriate technique. Liquid nitrogen has a slower cooling rate than dry-ice chilled isopentane, and is therefore more suitable for smaller tissue samples. If the tissue is considered fragile and less robust to fast cooling rates, an alternative that can be used for large samples is isopropyl alcohol, which slows down the cooling rate [14]. The samples will be cooled down to $-80°C$, at which they are stored until sequential analysis like IMS. Tissue samples can be stored at this temperature for at least a year before significant degradation occurs [16].

Formalin fixed paraffin-embedded (FFPE) tissue tries to prevent degradation and preserve sample integrity, using a formalin fixation process which bounds to the sample and prevents ionization [17]. Then the sample has to be decellularized, which means to remove cells from the extracellular matrix (ECM) scaffold. This processes improves the signal of ECM [18]. Next, the sample can be sectioned in thin slices through a method called cryo-sectioning. The slices are placed into a drying system (e.g. desiccator box) until they are used for IMS analysis [1].

Cryo-sectioning Cryo-sectioning describes the part in the preparation pipeline where a thin section with a thickness of around $10 - 20$ µm will be dissected of the original sample [14]. Once separated, the slices are thaw mounted onto microscope slides (e.g. fixing the sliced section to the sample plate) at a temperature of $-20°C$. The microscope slides are made out of conductive material like indium tin oxide coated glass [19]. Prior to thaw mounting, an alternative approach is to stretch the sample homogeneously in two dimensions, spreading out the sample over a larger surface for an increase in spatial resolution during IMS [19].

2-1-1-2 Tissue sampling

Multiple procedures exist to acquire IMS data sets: (i) synthetic data created by programs like MATLAB or Python, (ii) sampling the tissue samples by the user through ionization or (iii) using already existing databases (e.g. libraries which consist of well documented 2-D and 3-D IMS data sets [20]). This section solely focuses on ionization of the sample and subsequently obtaining the mass spectra.

Ionization takes place to make sure that the ions are extracted from the tissue and can be detected by a analyzer in order to obtain the mass spectrum of each individual pixel. Combining all pixels together, and taking their x and y coordinates into account, a complete image can be made with the use of this IMS data. The rest of this section will dive more into one specific ionization technique called Matrix Assisted Laser Desorption/Ionisation (MALDI). Several others exist like Desorption electrospray ionization (DESI) [21, 22, 23] and SIMS [24], but will not be discussed here.

Matrix Assisted Laser Desorption/Ionisation (MALDI) First usage of MALDI has been reported in 1984, but the technique gained more attraction through the work of Caprioli et al. in 1997 [25]. The popularity mainly manifested because of its high sensitivity of ion detection and ability to ionise a wide range of molecular weights and species, especially for peptides and proteins [26, 27]. MALDI-MS measures the chemical content in the form of a mass spectra at discrete spatial points, effectively repeating this process for each pixels in the image, creating a hyperspectral image [28]. A typical MALDI process can be seen in Figure 2-2.

After sample preparation, a chemical matrix will be deposited onto the tissue sample. This allows ionization of the analytes in the sample surface. Using laser pulses each between $100 - 200$ µJ in a high vacuum (HV) or ultra-high vacuum (UHV) environment, the tissue sample will be measured at these discrete points that are predefined [19]. The spatial resolution depends heavily on the minimization of the laser diameter, which can be achieved by changing the instrument's geometry [1]. Typical spot sizes of the laser are in the range of $5 - 300$ µm

[19]. With a m/z range of up to 500 kDa, MALDI is able to easily ionise larger molecules like proteins.

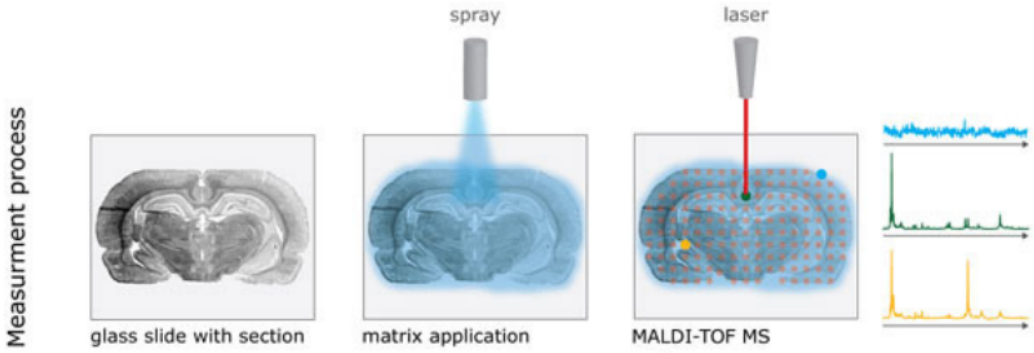


Figure 2-2: Visualisation of a typical MALDI-MS process. The application of the chemical matrix can be seen together with the placement of the discrete spatial points determining the pixels of the IMS image. Each pixels contains its own mass spectrum [28].

Since MALDI is label-free technology, it is possible to investigate a drug and its metabolites during one and the same experiment [29]. MALDI is also able to detect lipids, as long as it was prepared using the fresh frozen protocol, since other preservation processes deplete the lipid population [29]. The property of being a label-free technique makes MALDI also useful for discovery studies [28]. To that end, Balluff et al. successfully predicted the gene HER2-status, a protein responsible for regulating the growth of breast cells, obtaining more insight in breast cancer growth [30].

There exist other forms of MALDI, but not as widely implemented as regular MALDI. Some examples are scanning microprobe MALDI (SMALDI), infrared MALDESI (IR-MALDESI) and surface-assisted laser desorption/ionization (SALDI). As Schwamborn et al. point out that MALDI does have a lot of potential to be distributed more broadly in a clinical setting, but common sample preparation and measurement strategies between different laboratories are not yet established [31].

Matrix selection/ deposition MALDI requires a matrix in order for proper ionization, which slightly limits its applicability to larger proteins [1]. This matrix allows for ionization with the analytes at the tissue surface, and usually consist of an organic solvent with two functions. Firstly, analytes from the sample will be extracted using the organic solvent, and then co-crystallize with the matrix compound [20]. Another function is that the matrix helps to softly dissipate the energy coming from the laser to desorb and ionize intact analytes from the surface of the sample [19, 20]. Furthermore, the matrix should absorb UV light at the appropriate laser wavelength, sublimation should not be possible under HV or UHV conditions and the solubility of the matrix should be at the same level as the solvent and analytes [14].

Choosing the correct matrix depends on the required application and what the researcher wants to investigate. Typical matrices that are being used in pharmaceutical applications are 2,5-dihydroxybenzoic acid (DHB), α-cyano-4-hydroxycinnamic acid (CHCA), 9-aminoacridine (9AA) and 1,5-diaminonapthalene (DAN), because of their ability to extract and ionise small endogenous metabolites, drugs and lipids [14].

When applying the chemical matrix to the tissue sample, one should strive for homogeneous application of the matrix, document the application steps as good as possible, provide sufficient sensitivity and should be easy to use [19]. An heterogeneous application of the matrix would lead to locally differences when measuring the mass spectra over the whole image. These local differences are hard to detect. A well written documentation of the application of the matrix will increase the reproducibility of the process, which will ultimately lead to better comparisons by other laboratories. Interpretation errors will occur when multiple laboratories use different protocols for preparation of the matrix [32]. On top of that, the size of the matrix crystals determine the maximum spatial resolution that can be obtained [19]. Typical matrix crystal sizes lay between 5 and 150 μm, and are mostly depended on the deposition methods [14].

Several matrix deposition methods exist. Amstalden et al. mention briefly the principal methods for matrix deposition methods in IMS: dried-droplet method, pneumatic nebulization, chemical inkjet printer, automated vibrational spray coater, matrix sublimation and dry-coating [19]. Swales et al. recognises these methods, and differentiate them as 'wet' and 'dry' methods [14]. Wet methods are dried-droplet method, pneumatic nebulization, chemical inkjet printer and automated vibrational spray coater. These methods are being called wet, since the matrix is applied in liquid state. Dry methods like matrix sublimation and dry coating are possible as well, but the absence of liquid interface does limit the extraction of analytes from the tissue surface into the matrix crystals [14]. Manual techniques like dried-droplet method have the preference when experimenting, whereas more automated techniques are favorable when scaling up the matrix deposition phase, reducing the room for human errors.

The matrix can be sprayed or brushed upon the tissue sample using instructions from the manufacturer of the matrix, which are often quite detailed in order to make the whole process better reproducible [15]. McDonnell and Heeren point out that experiments with a high spatial resolution are more sensitive to analyte redistribution during the step of applying the matrix [33]. This is one of the reasons why the laser diameter will be relatively large, since then chemical information is superior compared to using a smaller laser diameter [33].

Removing the matrix is possible by washing the tissue sample with a 70% ethanol solution in order to dissolve the deposited matrix. Staining the remainder allows inspection by an optical microscope and ensures accurate co-registration [14].

2-1-1-3 Detection (mass analyzer)

Detection by the mass analyzer can be achieved using various instruments. According to Rubakhin et al. [13], there are at least four types of mass analyzers that are being used in IMS: (i) Time-of-Flight (TOF), (ii) Fourier transform ion cyclotron resonance (FTICR) mass spectrometer, (iii) magnetic/electric sector, and (iv) quadrupole ion trap. These days, mass analyzers using the magnetic/electric sector are less used than TOF instruments. Combining multiple mass analyzers will create an hybrid or tandem mass analyzer, which can often be found in commercially available IMS instruments today. Prior to choosing the correct mass analyzer, it is important to have incorporated high mass accuracy to allow confident identification of the desired molecules [1]. This subsection expands on TOF and FTICR instruments only.

Time-of-Flight (TOF) A detector broadly used are Time-of-Flight (TOF) based instruments. In essence, the simplest form of TOF mass spectrometers consist of an ion source (e.g. tissue sample, from which the ions are extracted) which will be directed into a tube via an electromagnetic field, and are being detected by the collector at the and of the tube [34]. See Figure 2-3 for an schematic overview of two possible modes for Time-of-Flight (TOF): linear and reflector mode [35]. In linear mode, the extracted ions have free path between the tissue sample and detector, whereas in the reflector mode the ions are deflected by ion mirrors before detected, effectively creating a longer traveled path for the ion. By creating a longer path for the ions to travel, distance between heavy and light ions will be magnified, thus increasing the resolution of the mass analyzer [35].

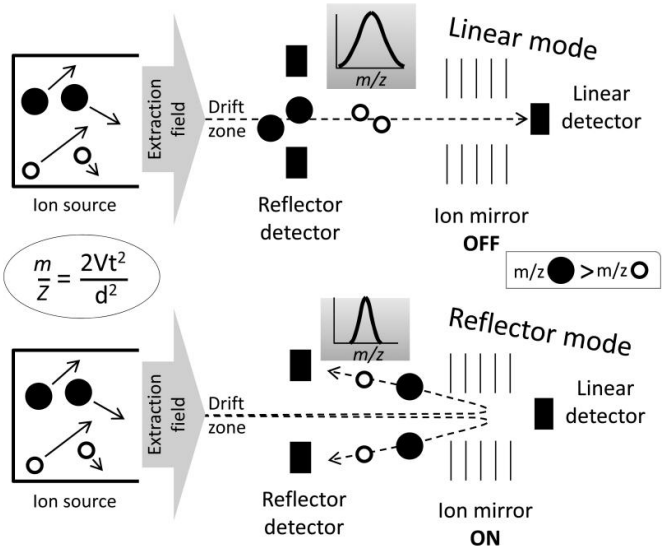


Figure 2-3: Two modes of the Time-of-Flight (TOF) mass analyzer: linear and reflector mode [35].

The potential of the electromagnetic field is known. When working out the energy balance equation, since the potential energy($E_p = zU$) will be converted into kinetic energy ($E_k = \frac{1}{2}mv^2$) the following relation can be obtained:

$$\begin{aligned} zU &= \frac{1}{2}mv^2 \\ zU &= \frac{md^2}{2t^2} \\ 2t^2 &= \frac{md^2}{zU} \\ t &= \sqrt{\frac{md^2}{2zU}} \end{aligned} \tag{2-1}$$

with t the time-of-flight, m the mass of the ion, z the corresponding charge, d the distance between the tissue sample and the detector and U the potential of the electromagnetic field. In this equation, the distance d and the potential of the electromagnetic field U are know, consequently the relationship between the time-of-flight t and the mass-over-charge m/z can be found:

$$t \propto \sqrt{\frac{m}{z}} \tag{2-2}$$

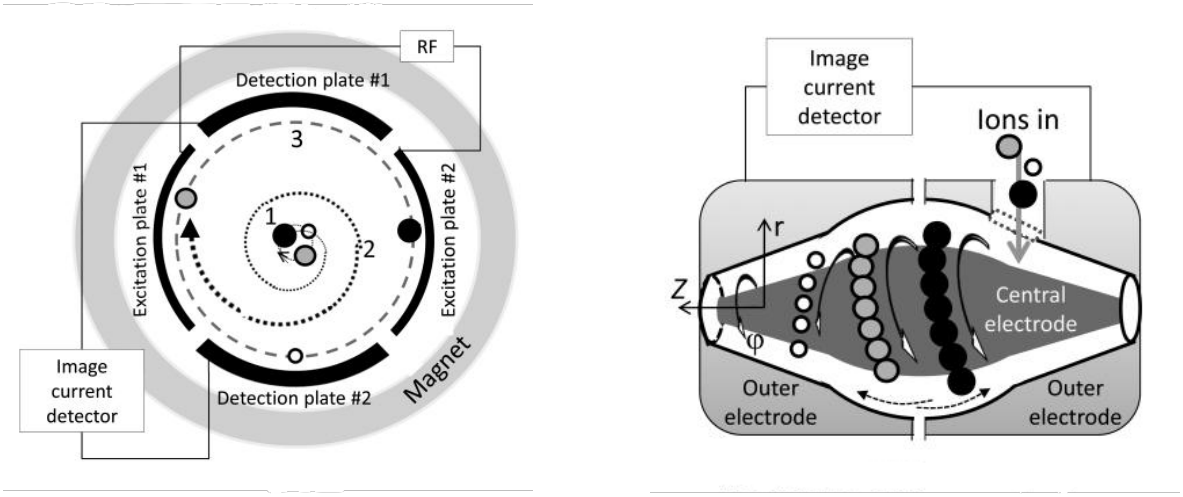
This would mean that the m/z is quadratically dependent on the time-of-flight t [34]. Heavier ions reach lower speeds, and thus have a larger time-of-flight, resulting in a higher m/z value. TOF-MS has the ability to detect ions from a broad mass range of maximal m/z 100 kDa, making it ideal for IMS [19]. The sensitivity of a TOF mass analyzer can be increased by elongating the travelled path of the ions d, effectively causing a larger separation between heavier and lighter ions.

Fourier transform ion cyclotron resonance (FTICR) ion cyclotron resonance (ICR) uses the rotational orbital frequencies of the extracted ions to separate the the heavier ions from the lighter ions. A combination of magnetic and electric field makes this separation possible [35]. As can be seen in Figure 2-4a, all ions start in the middle of the mass analyzer, and by applying a sequence of RF pulses to the excitation plates, ions with a certain m/z value can be separated by the rest of the ions and are being detected by the detection plates. An mass spectrum can be extracted by the mass analyzer by performing a thorough sweep across a broad range of different frequencies, and applying a Fourier Transform to determine the corresponding m/z at certain frequencies using the following equation [35]:

$$\omega = \frac{zB}{2\pi m} \tag{2-3}$$

with m the mass of the ion, z the corresponding charge, B the strength of the magnetic field and ω the rotational orbital frequency.

Another method of using a ICR-type instrument like the Orbitrap [36], which is more compact and operates slightly different compared to regular ICR (see Figure 2-4b). Ions are orbiting around a central electrode, and the frequency in which the the ions are oscillating around this electrode is related to the molecular weight in the following way:



(a) **(b)**

Figure 2-4: Two types of mass analyzer encorporating an FTICR technique: **(a)** A standard setup using ion cyclotron resonance (ICR), and **(b)** the more compact version applying ICR Orbitrap [35].

$$\omega = \sqrt{\frac{kz}{m}} \tag{2-4}$$

In this equation only m/z are unknown, and they can be easily computed by measuring the field curvature k and angular frequency ω, depending on what position of the detector the ions land.

Afterwards, a Fourier Transform is performed to obtain the final mass spectrum. FTICR systems are known for the highest obtainable resolution, scaling linear with the strength of the magnets being used within the mass analyzer [35].

2-1-1-4 Preprocessing

Once the samples have been scanned by the MS instrument, and the raw IMS data is obtained, several preprocessing steps can be taken before data processing and visualisation, in order to remove any flaws and biases in the measurements. Steps in the prepreocessing step may include normalization, baseline correction, spectra recalibration, smoothing, and data compression (unsupervised and supervised) [37]. All steps are optional, although normalization is expected to be incorporated into data analysis [1]. These preprocessing steps help reducing the variance within the data set. This section will go over several scaling and normalization methods, as well as common techniques like spectral alignment and peak-picking.

Scaling Scaling can be confused with normalization. Scaling changes the whole range of the data (e.g. all pixels at a specific m/z bin), while normalization changes the shape of the distribution of the data (e.g. the entire mass spectrum at a specific pixel). Scaling usually takes place after any other preprocessing step, as a final step before downstream analysis.

Wolski et al. found that the distance measure, the intensity scaling and their interactions mostly influenced the intensity based pairwise peak-list comparison [38]. They also found that the combination of the Euclidean distance with vector norm scaling, the Manhattan distance with Total Ion Count (TIC) and the sum of agreeing intensities with vector length scaling produced the best results [38]. This was tested on both Positive Matrix Factorization (PMF) and MS data sets.

Several scaling methods exist as pointed out by Verbeeck et al. [3]. A few examples are (i) Auto-scaling [39], (ii) Poisson scaling, (iii) Filter Scaling and shift variance scaling and (iv) intensity scaling to incorporate prior knowledge. It depends on the spectrum what type of scaling should by used, but for example Auto-scaling suffers from the fact that noisy low-signal m/z bins can have an increased impact on downstream analysis compared to the other more prominent signals [3]. For data of inherently Poisson nature, it is better to scale the data using Poisson scaling opposed to Auto-scaling before applying Principal Component Analysis (PCA), as the result may be misinterpreted otherwise [40]. This is due to the way PCA operates; the algorithm assumes that the data is normally distributed, which is not usually the case with real-world data. PCA will be further discussed in section 2-2-2

Normalization Choosing the correct normalization algorithm is crucial, since in certain cases some normalization algorithms may lead to possibly wrong conclusions, for example about the potential biomarker distributions [41]. Deininger et al. stress that two cases have to be considered in which the user can influence the intensity of signals in MALDI data sets. On a local level, e.g. at specific m/z bins, specific ion suppression can happen. The two different types of normalization distinguished by Vallejos et al. are (i) within-sample normalization and (ii) between-sample normalization [42]. These types of normalization are capable of removing intensity biases and adjust for small differences in for example the read counts between measurements [42].

In this section, several forms of normalization will be discussed and their effect on real-life data sets. The approaches in this section that will be discussed are normalization on Total Ion Count (TIC), vector norm, median and by noise level. These effects can occur due to several reasons, for example applying the matrix incorrectly, depth differences of ionization or gene-specific biases due to adjusted setting of the IMS instrument [1, 32]. In general a mass spectrum can be seen as a vector s with intensity values y_i [41].

$$\vec{s} = y_1, y_2, \cdots, y_n \tag{2-5}$$

Normalization is nothing more than dividing these intensity values by a factor f. This factor is different for other normalization techniques. The normalized mass spectrum then takes the following form:

$$\vec{s}_{\text{normalized}} = \frac{1}{f}\vec{s} \tag{2-6}$$

Total Ion Count (TIC) The most commonly implemented method is the TIC normalization. The normalization on TIC uses the so called p-norm with p=1, which has the following form:

$$f = \left(\sum_i |y_i|^p \right)^{\frac{1}{p}} = \sum_i |y_i| \tag{2-7}$$

The normalization will be based on the sum of all intensities in the mass spectrum [41]. Substituting this value for p into Equation 2-7, and then into Equation 2-6 to obtain the normalized mass spectrum for TIC gives the following:

$$\vec{s}_{\text{TIC}} = \frac{1}{\sum_i |y_i|} \vec{s} \tag{2-8}$$

TIC normalization is the most effective at recovering the linear behaviour of the spectrum, and increasing the correlation coefficient [37]. It however is based on the assumption that the total amount of signals is comparable in each mass spectrum. When this assumption is met, the technique is easy to use and proven to be powerful.

Excluding mass ranges Prior to applying TIC and vector norm, manual adjusted mass ranges can be excluded from the normalization [41]. This consideration depends on the region of interest to the user. The intensities within a region are set to zero, such that they will be ignored by the normalization algorithm. Applying this technique does expect supervision from an expert, which is an extra step in the process and time consuming. However, Deininger et al. found out that TIC with manual mass exclusion produced the best results [41]. TIC normalization remains the popular option for normalization in IMS, but it is proposed to use normalization on the median to spot artifacts caused by TIC normalization [41]. A visual comparison of the normalization techniques which are partly discussed in this subsection can be seen in Figure 2-5.

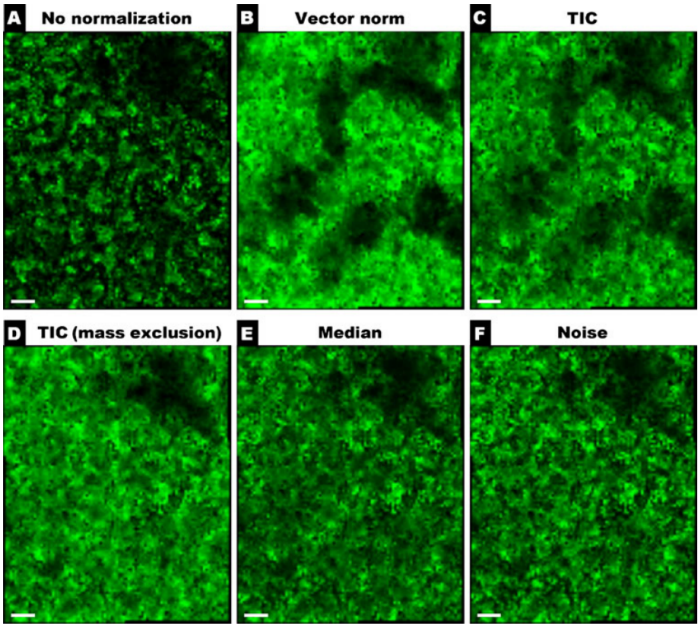
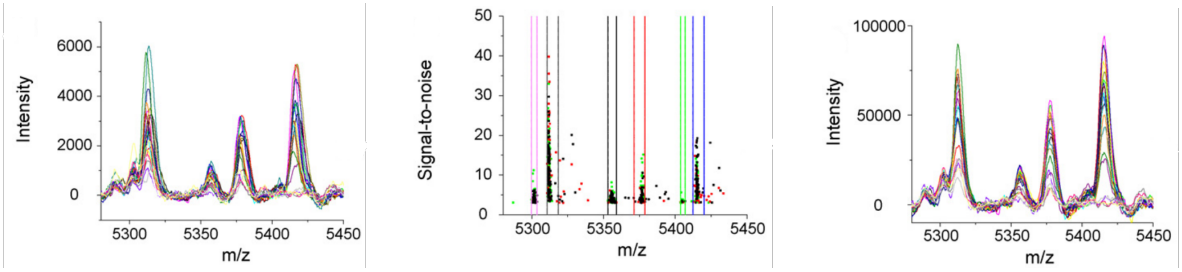


Figure 2-5: Comparison of several normalization techniques. **(A)** No normalization, **(B)** vector norm, **(C)** TIC, **(D)** TIC (mass exclusion), **(E)** normalization on the median and **(F)** normalization by noise level [41].

Spectral realignment Before a fair comparison of a certain m/z value takes place, it is necessary to be sure that the intensities of the m/z bins is correct over all measured mass spectra [37]. The effect of an misaligned spectrum is less problematic for a spectrum with wide peaks, compared to thin and sharp m/z peaks. Spectral realignment tool are able to adjust for small deviations in the mass spectra between different pixels by shifting the whole spectrum slightly to the left/right depending on the peak locations [36]. Tracy et al. used a maximum likelihood method to detect peaks in the mass spectra, process the list of detected peaks and alignment of the original peak list [43]. It was found that the precision of the mass spectral data can be increased using this method.



(a) Raw mass spectra **(b)** Binned peaks **(c)** Realigned mass spectra

Figure 2-6: Spectral realignment process. **(a)** Original raw mass spectra, **(b)** the binned peaks from the data set (categorized into bins, grouping peaks from similar ions), and **(c)** the resulting realigned mass spectra [37].

Peak-picking Peak-picking is can simply described by filtering the mass spectrum and registering the most prominent peaks, converting the mass spectrum to a discrete set of peaks [4], removing the other signals below a certain chosen threshold. Peak-picking is not necessarily a similar preprocessing technique as all the other techniques mentioned in this section so far. It essentially is already a form of an dimensionality reduction method, a feature selection algorithm to be exact (more on this in section 2-2). The IMS data will be compressed, therefore causing information loss by throwing away the signals that did not reach the chosen threshold [44]. A visualisation of a peak-picking algorithm can be seen in Figure 2-7.

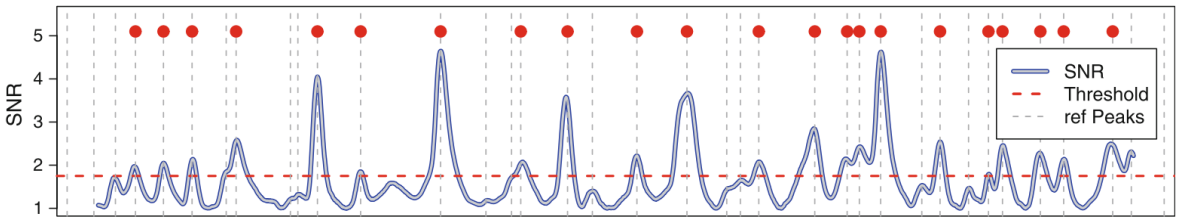
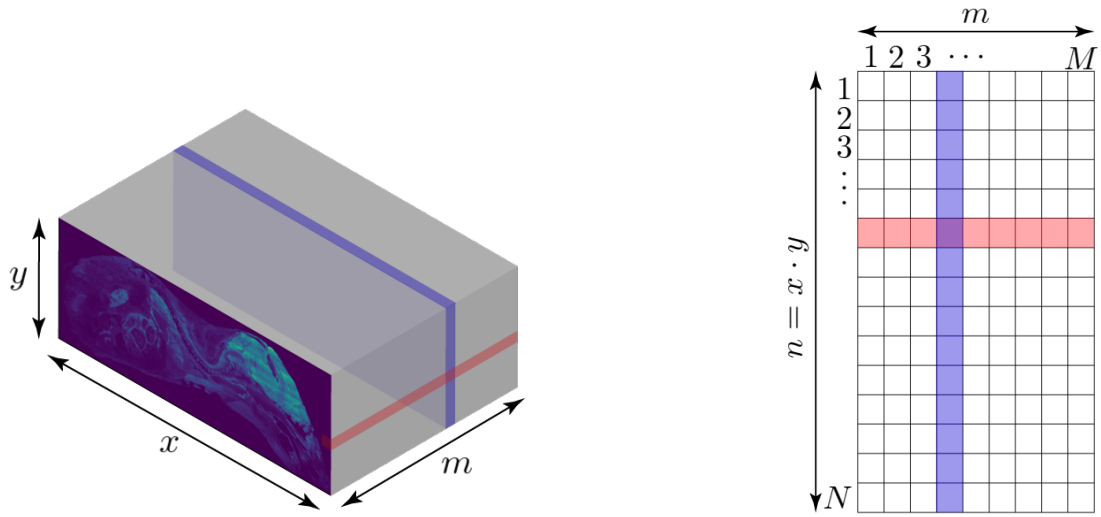


Figure 2-7: Example of a peak-picking algorithm. Peaks above a certain threshold are being detected, annotated with red dots [45].

2-1-2 Structure of IMS data sets

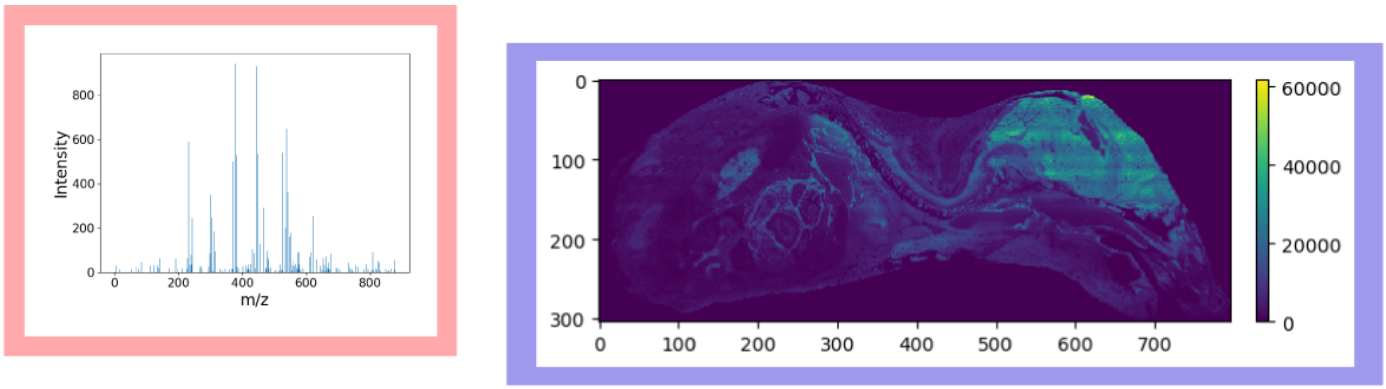
As mentioned before, the IMS will be stored into a hyperspectral image or datacube [28]. A visualisation of this 3-D datacube can be seen in Figure 2-8a. All pixels are stored in x and y, where each pixels contains a mass spectrum which is stored in m, with a total number of spectra M. Slicing this 3-D datacube orthogonal to the m axis (blue) gives a ion image, see Figure 2-9b. Each pixel contains a mass spectrum (red), which can be seen in Figure 2-9a. Prior to downstream analysis, this 3-D datacube is often flattened into a 2-D matrix (Figure 2-8b). All pixels in x and y direction are stored underneath each other in n, with a total number of pixels N.



(a) 3-D datacube **(b)** 2-D matrix

Figure 2-8: Visualisation of the hyperspectral image or datacube in which the IMS data is stored. **(a)** shows the 3-D datacube. The pixels in x and y direction are stored, as well as their m/z values in m. Ion images can be seen as the slices (blue) of this datacube, and the mass spectrum (red) for every pixel of all images combined. **(b)** Flattening the 3-D datacube creates a matrix of two dimensions, storing at total of N pixels in n, each containing a total of M mass spectra in m. The ion image and mass spectrum can be distinguished by extracting a column (blue) or row (red) from this 2-D matrix respectively. This representation is often be used for sequential analysis instead of the datacube.

Technological improvements cause IMS data sets to expand throughout the years. The main factors that influence the data size are depicted in Table 2-1. Extending the m/z range or resolution only scales linear, since only one dimension of the 3-D datacube in Figure 2-8a increases in size [4]. Increasing the spatial resolution (e.g. decreasing the laser diameter used in MALDI or extending the measured tissue surface) has a quadratic impact on the data size, since the 3-D datacube increases in both x and y dimension. The precision at which the ion counts are stored is usually a few bytes, but depends on the data acquisition instrument. Since this property is connected to the m/z value, increasing the precision would result in linear scaling of the IMS data size. Raw IMS data can easily get in to GBs for a single measurement, resulting in data containing 100k$-$500k pixels each with $10^4 - 10^6$ m/z bins [3]. Further technological advancements in the IMS field result in even larger data sets.



(a) Mass spectrum **(b)** Ion image

Figure 2-9: Example of a mass spectrum and an ion image following the similar colors in Figure 2-8. **(a)** An typical mass spectrum, showing all detected m/z values with arbitrary intensity. **(b)** An ion image shows an false-color image of a certain m/z value.

	Factor	Scaling
1	Extent of the scanned mass range	Linear
2	m/z resolution (m/z bin size)	Linear
3	Extent of measured tissue surface area	Quadratic
4	Spatial resolution (pixel size)	Quadratic
5	Precision with which the ion counts are stored	Linear

Table 2-1: Main factors influencing the IMS data size [4].

2-2 Dimensionality Reduction

IMS data sets have scaled throughout the years through advancements on a technological level, causing an urgent need for better visual understanding [1]. Raw IMS data can easily reach a few terabytes for a single experiment; a datacube containing $10^3 - 10^6$ pixels each with $10^4 - 10^6$ m/z bins [3]. High-dimensional data can lead to problems; storage being the first obvious obstacle. Preferably, the size of the datacube would be as minimal as possible to fit entirely on the memory of the workstation. Subsubsection 2-2-1-1 will go more into detail of the problems depending on data size regarding to storage, and subsubsection 2-2-1-2 will discuss the curse of dimensionality. This section goes over the motivation of applying dimensionality reduction, as well as the different methods in which this LD embedding can be obtained.

Dimensionality reduction seeks a way to reduce the number of features of the data, while trying to retain as much information of the original data as possible. In the previous section, the structure of IMS data is explained. Considering the two dimensional matrix of Figure 2-8b, this high-dimensional data contains N measurements (pixels) with each M features (m/z bins). A feature can be described as an individual measurable property of an observable process [46]. The 2-D matrix can be mathematically described as being of dimension $\mathbb{R}^{N \times M}$

with the number of pixels N and number of dimensions (m/z bins) M. Dimensionality reduction compresses the data along the "features" axis, reducing the data to $\mathbb{R}^{N\times K}$ with $K < M$. This is visually displayed in Figure 2-10.

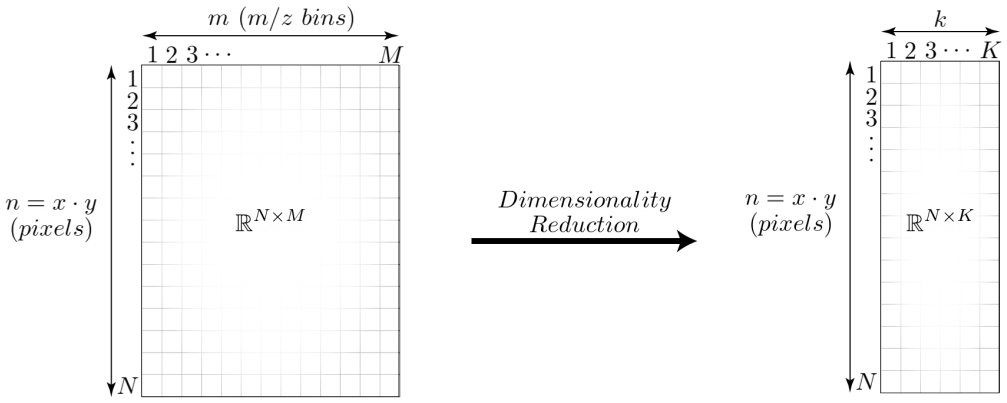


Figure 2-10: Visualisation of dimensionality reduction. The number of data points (pixels) N remain constant, yet the number of features (m/z bins) will be reduced from M to K.

Reduction is useful for several reasons. Depending on the application, often a low-dimensional structure is hidden within the high-dimensional data [47]. Belkin and Niyogi expand on this statement with an example of gray scale images of an object taken by a camera under fixed lighting conditions. While each image would contain N pixels, resulting in high-dimensional data of $\mathbb{R}^N$, the intrinsic dimensionality of the space of all images of the same object is only determined by the degree of freedom of the camera [47].

In terms of IMS data, it would mean that an underlying structure can be found of size K, which is lower than the number of m/z bins M. This section will try to expand on several other benefits of dimensionality reduction.

Feature selection Feature selection can be described as a selection process which efficiently extracts an optimal subset of features from the original high-dimensional data while suppressing any noise or irrelevant noise within the data [48]. The criterion of this selection process is required to determine whether a certain feature can be added to the optimal subset of features. Chandrashekar and Sahin mainly review three supervised feature selection methods (e.g. applied in classification problems); (i) Filter methods, (ii) Wrapper methods and (iii) Embedded methods [46].

Filter methods review each feature separately by only taking the data into account, using statistical measures like the Pearson correlation coefficient or information theoretic ranking criteria [46]. Filter methods are relatively computationally light, do not rely on learning algorithms and avoids overfitting. Wrapper methods are more sophisticated, meaning there is already some sort of predictor built into the method that is being used as a black-box for classification. The predictor performance is being used as selection criterion and evaluation of the variable subset of features [46]. Using either sequential or heuristic search algorithms the sub-optimal subset can be found by determining the global maximum of the used predictor. A drawback of this method is the increase in computation time due to reclassifying different subsets. Embedded methods try to resolve this bottleneck by incorporating the feature

selection process within the training process [46].

Opposed to supervised learning dimensionality reduction, there also exist examples of semi-supervised learning and unsupervised learning methods in the literature. Where supervised learning uses labels for its classification problems, semi-supervised learning uses labels only partly during the duration of the algorithm. Unsupervised learning uses unlabelled data through which the algorithm tries to find the hidden structure within the data by applying for example clustering techniques to classify the data points [46]. In terms of IMS data, unsupervised learning deals with the problem that the biological meaning of the m/z values is unknown. Purely through data mining the aim is to reveal general data structure [28]. Similar to IMS, scRNA-seq data deals with high dimensionality as well, hence Andrews and Hemberg applied feature selection to identify the underlying structure [49].

Feature extraction Besides feature selection, the field of feature extraction tries to reduce the number of features by fusing features together by their most prominent properties. Similar to feature selection, the high-dimensional of size $\mathbb{R}^{N\times M}$ will be reduced by feature extraction to $\mathbb{R}^{N\times K}$ with $K < M$. Although now k contains not just a subset of the original features, but the LD embedding has been transformed and is a new representation of the original data. The features of the LD embedding can be a linear or nonlinear combination of the high-dimensional features. The following part of this chapter will elaborate on the different methods constructing this low-dimensional feature space in a linear or nonlinear procedure. Another approach of classifying the different feature extraction methods would be methods that mostly preserve global distance structure versus methods that maintain local structure over global structure within the data [11].

2-2-1 Motivation

2-2-1-1 Size of the data

It has been mentioned that the raw IMS data can be stored into a datacube containing $10^3 - 10^6$ pixels each with $10^4 - 10^6$ m/z bins [3]. In subsection 2-1-2 the main factors determining the size of the data size were discussed. Since the underlying structure of the tissue sample most possibly will be smaller than the number of measured m/z bins, "compressing" the data would result in less necessary storage for the datacube while retaining only the most valuable information. Determining what part of the original data should be retained in a lower-dimensional embedding is decided my dimensionality reduction techniques. For downstream analysis, the whole data set is usually stored into the available RAM of the workstation. With less dimensions to consider, further calculations would benefit from a substantial decrease in runtime.

2-2-1-2 Curse of dimensionality

With increasing dimensionality of a data set, the volume of the space increases exponentially resulting the variance in the distances between the data points to approach zero [44]. In terms of IMS, the intended space is built up with each m/z being its own dimension. For IMS data sets consist of thousands of m/z bins, large dimensions are reached for the original data. The

reduction of the variance towards zero is referred to as the curse of dimensionality; especially the Euclidean distance suffers from this effect [3]. Palmer et al. also argues that in order to evaluate data properly, the necessary needed number of data points grows exponentially with the number of dimensions. In high-dimensional space the data gets sparse, and traditional algorithms which requires the distance metric between data points drop in efficiency and/or effectiveness [50]. The sparsity is visible in Figure 2-11, visualizing the increased distance between the same number of data points in higher dimensions. Scaling the dimensionality of the data even further only creates more sparsity in the data. One should limit analysis to lower dimensions due to this curse of dimensionality.

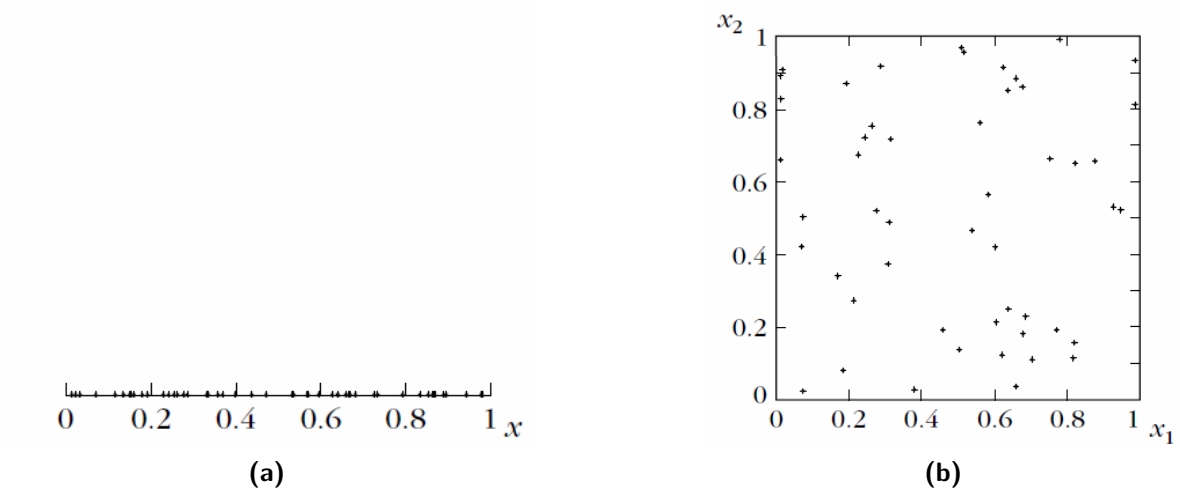


Figure 2-11: Visualisation of the sparsity of high-dimensional data. Fifty data points have been generated by a uniform distribution, plotted in **(a)** one dimension and **(b)** two dimensions. Clearly the distances between the data points increases when scaling the dimensions, introducing the concept of sparsity in the data [51].

2-2-2 Linear decomposition methods

Linear dimensionality reduction methods seek for a linear mapping from the high-dimensional space to the LD embedding, by minimizing the explained variance of the data points. Mostly this is done by matrix factorization. Various methods exist, like Principal Component Analysis (PCA) [52, 5], Independent Component Analysis (ICA) [53], Maximum Autocorrelation Factorization (MAF) [54] and Non-negative Matrix Factorization (NMF) [55]. While these methods have been used extensively, PCA has been applied most frequently, and is shortly described in this section. Matrix factorizations are being widely used for a long time already, and have been optimized in every common programming language (e.g MATLAB, Python).

An advantage of linear dimensionality reduction methods is the explainability of the LD embedding. For PCA it holds that the axis show proportional how much explained variance can be found in either direction of the axes [5]. Nonlinear dimensionality reduction methods sand their interpretability will be discussed in subsection 2-2-3.

Principal Component Analysis (PCA)

PCA [52, 5] is the most commonly known form of a linear Dimension Reduction (DR) technique, and is actively used as a processing step in linearly reducing the number of dimensions [56, 57]. PCA aims to extract a new representation in a reduced number of features from the high-dimensional data that maximizes the variance in the low-dimensional space. Visually, PCA searches for a best fitted straight line through the data points, minimizing the variance of all the data points w.r.t. that line [5], as can be seen in Figure 2-12. When this fit is found, the next fit is constructed *orthogonal* to the last line using the same algorithm. This process is repeated until the desired number of dimensions K is reached, with $K < M$. These lines represent the directions of the eigenvectors of the principle components (PCs), which are ranked by importance.

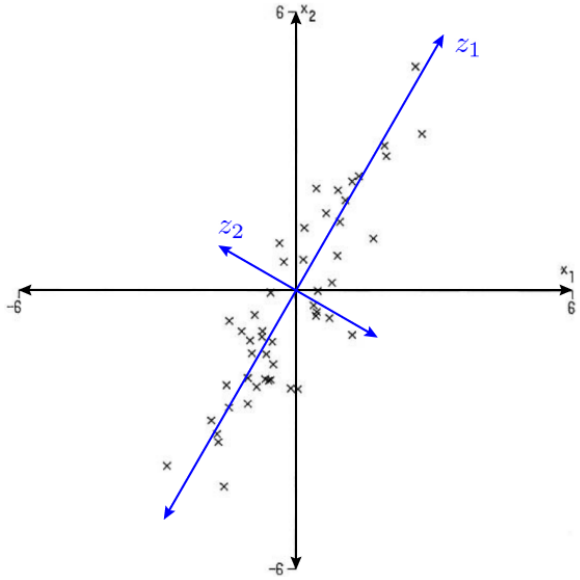


Figure 2-12: Construction of the PCs in a cluster of data points. The original axis are labeled as x_1 and x_2. The direction of the eigenvectors, and also the first and second Principle Component, are depicted as z_1 and z_2, respectively [5].

These steps can be written in the form of a matrix decomposition. In order to do so, first the covariance matrix will be calculated.

$$\text{cov}(A, A) = \mathbf{E}\left[(A - \mu_A)^\top (A - \mu_A)\right] \tag{2-9}$$

With A the data matrix, and $\mu_A = \mathbf{E}[A]$. When the data matrix is zero mean and column centered, the expression of the covariance matrix simplifies to the following:

$$\text{cov}(A, A) = \frac{A^\top A}{n - 1} \tag{2-10}$$

With n the number of data points. The eigenvectors, in other words the *principle components (PCs)*, can be computed in two ways; (i) applying an eigenvalue decomposition or (ii) compute

the Singular Value Decomposition (SVD). Previous work has proven that SVD provides superior numerical stability over eigenvalue decomposition [58], as has also been pointed out by Verbeeck et al. [3]. The SVD can be performed for the data matrix A of dimensions $m \times m$ as follows:

$$A = U\Sigma V^\top \tag{2-11}$$

With $U \in m \times m$ being a unitary matrix ($UU^* = UU^\top = I$) containing the left singular values of A, Σ a matrix of size $m \times n$ containing all the singular values on its diagonal and $V^\top$ also an unitary matrix ($VV^* = VV^\top = I$) of size $n \times n$ containing the right singular values of A. Substituting Eq. (2-11) into Eq. (2-10) gives:

$$\begin{aligned} \text{cov}(A, A) &= \frac{A^\top A}{n-1} \\ &= \frac{V\Sigma U^\top U\Sigma V^\top}{n-1} \\ &= V\frac{\Sigma^2}{n-1}V^\top \end{aligned} \tag{2-12}$$

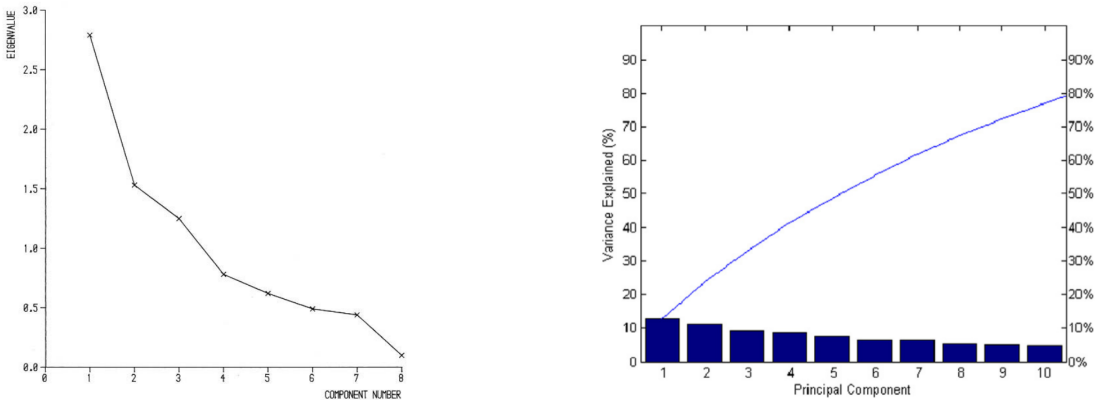
In this expression, the matrix V contains all the eigenvectors and $\frac{\Sigma^2}{n-1}$ all the corresponding eigenvalues, in decreasing order. These eigenvectors with corresponding eigenvalues are equivalent to the principle components ranked by importance. The desired number of dimensions d of the LD embedding are the first d principle components. Determining the right amount of PCs remains challenging [59]. To give more insight in the PCs, there are visual tools like the Scree plot and the Pareto plot (see also Figure 2-13). These plots show the value of the eigenvalue per PC and cumulative explained variance as a function of the number of components, respectively. Within these plots, thresholds can be set in order to select the amount of PCs with as objective a certain percentage of explained variance. In the literature, Deininger et al. were able to distinguish the tumor from the non-neoplastic mucosa using PCA with only the first 3 PCs, showcasing the useful applicability of PCA [60].

2-2-3 Nonlinear Dimensionality Reduction (NLDR)

A major drawback of linear decomposition methods like PCA is the inability of dealing with the nonlinear structure hidden within the data. In the Pareto plot of Figure 2-13b it can be seen that when using the only the first two PCs in order to plot the LD embedding, only 20% of the explained variance in the data have been accounted for. Roughly 80% of the structure within the data can not be visualized properly.

Nonlinear Dimensionality Reduction (NLDR) attempts to uncover the underlying nonlinear manifold of the data (if present at all), by constructing the low-dimensional feature space by variables that are nonlinear combination of the original features. A visualisation of how a NLDR unpacks the lower dimensional structure from the high dimensional data can be seen in Figure 2-14.

Although NLDR is superior in uncovering the underlying nonlinear manifold, if the data behaves linear locally, the subspace can be clearly identified with the use of linear techniques like PCA [3]. In order to find these nonlinear structures globally, several nonlinear



(a) **(b)**

Figure 2-13: Examples of 2 different visualisations describing the output from a PCA process: **(a)** the importance of the PCs in a Scree plot [5] and **(b)** displaying the cumulative explained variance in a Pareto plot [61].

manifold learning methods exist. Some examples of NLDR methods are: Sammon mapping [63], multidimensional scaling (MDS) [64], Self-Organizing Maps (SOMs) [65], Locally Linear Embedding (LLE) [62], Diffusion maps [66], Stochastic Neighbor Embedding (SNE) [7], t-Distributed Stochastic Neighbor Embedding (t-SNE) [6] [67], LargeVis [68], Uniform Manifold Approximation and Projection (UMAP) [11], Noise Contrastive Approach for Scalable Visualization (NCVis) [69] and many others [70, 71].

This subsection will go more into SNE and its successful successor t-SNE [6]. The latter has been the gold standard for NLDR techniques for the past decade. A few years ago UMAP has been introduced [11], showing already promising results which will be thoroughly discussed. Finally, the main comparison will be between the NLDR techniques t-SNE and UMAP.

Stochastic Neighbor Embedding (SNE)

SNE tries to optimally preserve local structure of the data by placing the similar data points closer together and dissimilar points farther away. SNE achieves this by using a probabilistic approach of mapping the data points from high-dimensional space to the LD embedding [7]. Mathematically, by centering a Gaussian on each data point i in the high-dimensional space a probability distribution p_{ij} can be calculated between i and neighbor j.

$$p_{j|i} = \frac{\exp\left(-d_{ij}^2\right)}{\sum_{k \neq i} \exp\left(-d_{ik}^2\right)} = \frac{\exp\left(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\left\|x_i - x_k\right\|^2 / 2\sigma_i^2\right)} \tag{2-13}$$

With d_{ij} the distance between point i and j in high-dimensional space, the variance of the Gaussian σ_i that is centered on data point x_i, and k the number of effective neighbors or the so called *perplexity*. In this expression the squared Euclidean distance function has already been substituted. The value for the perplexity can be set by hand by the user. The Gaussian probability σ_i is most likely not similar for each data point i, so this will be found using the perplexity.

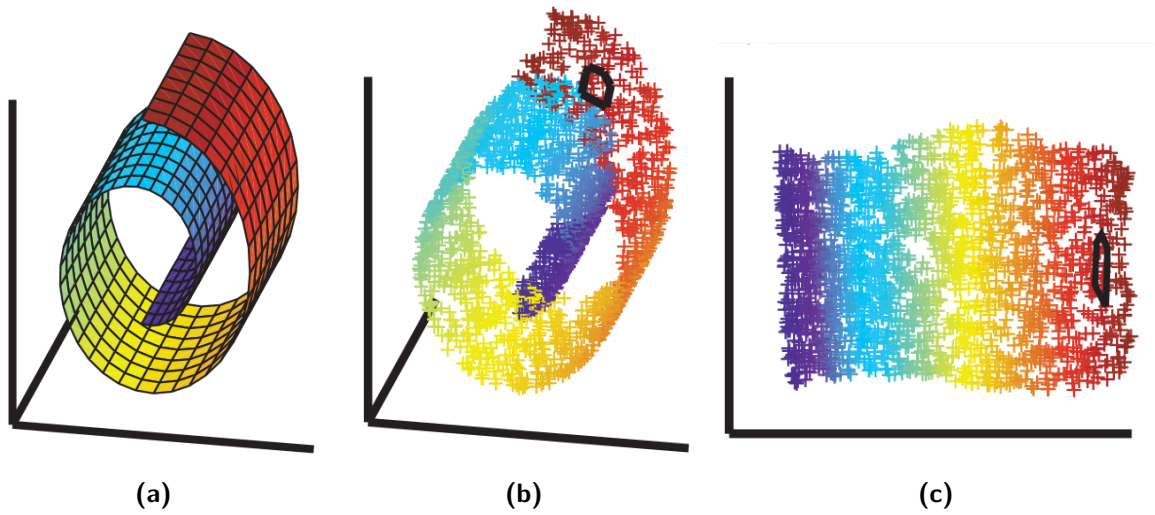


Figure 2-14: Visualisation of a Nonlinear Dimensionality Reduction (NLDR) technique called Locally Linear Embedding (LLE). **(a)** The original data set of a two dimensional plane within a three dimensional space, commonly known as a 'Swiss roll'. **(b)** This Swiss roll has been sampled to create the final input for the dimensionality reduction technique. **(c)** LLE is able to unfold the LD embedding (2-D) from the high-dimensional data (3-D) [62].

$$\text{Perplexity} = 2^{H(P_i)} = 2^{-\sum_j p_{j|i} \log_2 p_{j|i}} \tag{2-14}$$

With $H(P_i)$ being the Shannon entropy of P_i measured in bits [6]. Van der Maaten and Hinton argues that SNE is robust for different values of perplexity, and that typical values are somewhere between 5 and 50. For the LD embedding, a similar probability distribution can be obtained, and this time the variance of the Gaussian is set to $\sigma = \frac{1}{\sqrt{2}}$. A fixed value for σ only results in a rescaled version of the LD embedding.

$$q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y_i - y_k\|^2\right)} \tag{2-15}$$

With y_i and y_j the representations of x_i and x_j in the LD embedding. By now, 2 different probability distributions have been obtained: the probability in high-dimensional space $p_{j|i}$ and in the LD embedding $q_{j|i}$. p_{ii} and q_{ii} are set to zero, since only the pairwise interactions are interesting for the optimization problem. The whole goal of SNE is finding the embedding that represents the high-dimensional data the best. In this algorithm this is done by minimizing a cost function: the Kullback–Leibler divergence (KLD).

$$C^{SNE} = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} = \sum_i KL\left(P_i \| Q_i\right) \tag{2-16}$$

This cost function can be minimized using different methods. Steepest descent in which all the data points are being optimized simultaneously is inefficient and would get stuck in poor local minimum [7]. SNE minimizes the cost function using Gradient Descent (GD). In order to

use Gradient Descent (GD), the gradient is needed to calculate the slope of the cost function each step. Fortunately, this gradient has a relative simple expression.

$$\frac{\delta C^{SNE}}{\delta y_i} = 2\sum_j (y_i - y_j)\left(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j}\right) \tag{2-17}$$

The term $\left(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j}\right)$ can be interpreted as a spring system; an attractive or repellent force between the data points y_i and any y_j in low-dimensional space. The amount of force, and in which direction the force is applied, depends on the similarity of the two probability distributions $p_{j|i}$ and $q_{j|i}$. This force will be determined by the GD method. In order to not get stuck into poor local minima, Gaussian noise is added in the early stages of the optimization. The initialization of the LD embedding is random.

t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-Distributed Stochastic Neighbor Embedding (t-SNE) is based on its similar named predecessor SNE, but tries to overcome this "crowding problem" that emerges in data when being in high-dimensional space. The crowding problem entails the difficulty in faithfully displaying data points that are far away from each other at similar ratios compared to closer by data points. t-SNE tries to resolve this issue by using a Student t-distribution instead of a Gaussian distribution to compute the similarity between data points in low-dimensional space [6]. The difference between this two distributions can be seen in Figure 2-15. It basically comes down to the longer tails Student t-distribution, spreading out further away than the Gaussian distribution. Another modification that has been made is a symmetrization of the cost function, which has already been addressed by Cook et al. [72].

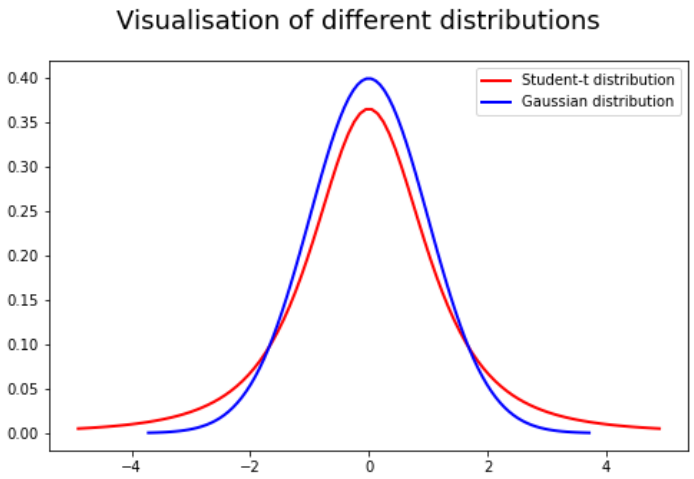


Figure 2-15: Visualisation of the Student t-distribution compared to the Gaussian distribution. Results are plotted using the `numpy` library in Python.

t-SNE adopts the same approach of minimizing the KLD, but now only a *single* KLD is computed between a joint probability distribution P and Q in respectively high and low-dimensional space, instead of every possible probability [6]. In this formula the approximate

expressions for the probabilities in high and low-dimensional space have been substituted; $P(X) \approx e^{-X^2}$ and $Q(Y) \approx \frac{1}{1+Y^2}$ respectively, followed by a derivation of Oskolkov [73].

$$\begin{aligned} C^{t-SNE} &= KL(P(X)\|Q(Y)) \\ &= \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \\ &= P(X) \log \left(\frac{P(X)}{Q(Y)} \right) \\ &= P(X) \log P(X) - P(X) \log Q(Y) \\ &\approx -P(X) \log Q(Y) \\ &= e^{-X^2} \log(1 + Y^2) \end{aligned} \tag{2-18}$$

The term $P(X) \log P(X)$ goes to zero for small and large distances between data points [73]. Similar to SNE, it holds that $p_{ii} = q_{ii} = 0$. Van der Maaten and Hinton refer to this form as symmetric SNE, and it has the property that $p_{ij} = p_{ji}$ and $q_{ij} = q_{ji}$ for $\forall i, j$ [6]. Then the probability distribution of the LD embedding is stated as follows:

$$q_{ij} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq l} \exp\left(-\|y_k - y_l\|^2\right)} \tag{2-19}$$

In order to handle outliers in high-dimensional space better, t-SNE uses a different probability distribution in low-dimensional space.

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{2-20}$$

With N the number of data points. As mentioned in the original paper, the main advantage of using this symmetric SNE is that the gradient gets simplified which results in a faster computation and a overall speed-up of the algorithm [6].

$$\frac{\delta C^{t-SNE}}{\delta y_i} = 4 \sum_j (y_i - y_j)(p_{ij} - q_{ij}) \tag{2-21}$$

Alternative versions of t-SNE The computational complexity of t-SNE is $O(N^2)$, which is far from ideal for large data [6]. Therefore, in the years after the invention of t-SNE, there have been many attempts in speeding up the algorithm. Barnes-Hut t-SNE is an example of that, which uses the Barnes-Hut algorithm [74] for approximating the forces between the data points in the embedding [8]. This decreases the computational complexity to $O(N \log N)$ which makes an huge improvements for large data sets, as is the case in the IMS field. Van der Maaten also compared Barnes-Hut t-SNE to a version of t-SNE which uses a dual-tree variant [75], but found that Barnes-Hut t-SNE only slightly outperforms dual-tree t-SNE due to additional bookkeeping (e.g. needing more memory) that is required in dual-tree t-SNE [8]. Linderman et al. managed to accelerate the algorithm even further with a version called

FIt-SNE [9]. A computational complexity of $O(\sim 2pN)$ has been achieved, with a small number p of interpolation nodes that control the interaction between two data points [9].

Hierarchical Stochastic Neighbor Embedding (HSNE) aims to bridge the gap between accuracy and interactivity, with a focus on reconstructing a representation of the hierarchical data [76]. HSNE has been specifically introduced for analysis on mass cytometry data sets [77]. Mass cytometry uses an inductively coupled plasma to ionize the tissue sample, causing the ions to be detected by a type of mass analyzer. Nonlinear similarities were successfully distinguished by HSNE, identifying rare cell populations that were previously missed due to downsampling [77].

A faster algorithm is always desirable, but good interpretation of the data is just as important. Determining appropriate parameters for a dimensionality reduction method remains difficult, since the algorithms are being influenced by the data. Different data sets would require a different parameter configuration. Opt-SNE tries to resolve this by automating the parameter selection of t-SNE by monitoring the cost function in real-time [12]. By utilizing the Kullback–Leibler divergence (KLD) evaluation, the early exaggeration phase can be calibrated. The early exaggeration phase of t-SNE will further be discussed in subsection 3-1-1. Although Belkina et al. showed that the final LD embedding results in a lower value for the KLD than the default settings of t-SNE, it is difficult to determine the correct value for the stopping criterion of the algorithm.

2-3 Distance metric

A distance metric is used to measure the distances between data points in the space they lie in. Clustering algorithms use these distances to compare the similarity between data points in order to assign them to a cluster, hence the applied distance metric is of extreme importance [3]. In IMS context, clustering algorithm aims to describe how similar or dissimilar the spectra of each pixel and thus its chemical content are to those of the other pixels [78]. Changing the way data points are described "similarly" by incorporating a certain distance metric, the conclusions based upon those clusters can vary. The most widely used metric is the Euclidean metric, but several others exist like Mahattan, Minkowski, Chebyshev, Cosine, Mahalanobis, Poincaré distance, Pearson correlation and many more. New distance metrics can even be created, like the Histomatch distance which has been applied to IMS data, since a distance metric is merely an equation describing the relationship between data points within a certain space [78]. Smets et al. found that Histomatch performs better than the Euclidean distance metric, and almost similar as the Cosine distance in terms of spatial autocorrelation [78]. Mathematically, the formulas of several distance metrics being used in data processing can be seen in Table 2-2.

Winderbaum et al. found that applying k-means clustering using the Cosine distance metric yield superior results as opposed to using the Euclidean distance metric [79]. The Cosine distance has also been used in UMAP to identifying hundreds of cell types and 56 trajectories of 61 embryos [80]. Collectively, thousands of corresponding marker genes were defined. Qian et al. have compared the Euclidean with the Cosine distance metric for nearest neighbour queries in high dimensional space, and did not found noticable advantages of using the Cosine distance over the Euclidean distance [81]. In another study, Sarkari et al. investigated k-means clustering using four different distance metrics: Euclidean, Cosine, Manhattan and

Distance Metric	Equation
Minkowski	$d(x,y) = \left(\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p\right)^{1/p}$
Euclidean	$d(x,y) = \sqrt{\sum_{i=1}^{n} \lvert x_i - y_i \rvert^2}$
Chebyshev	$d(x,y) = \max_i \left(\lvert x_i - y_i \rvert\right)$
Cosine	$d(x,y) = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$
Pearson Correlation	$d(x,y) = 1 - \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n} (y_i - y)^2}}$
Mahalanobis	$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}$ With S being the covariance matrix.
Poincaré	$d(x,y) = \operatorname{arcosh}\left(1 + 2\frac{\lVert x - y \rVert^2}{(1 - \lVert x \rVert^2)(1 - \lVert y \rVert^2)}\right)$
Histomatch [78]	$d(x,y) = 1 - \sum_{i=1}^{n} \max\left(0, \min\left(x_i - y_i\right)\right)$

Table 2-2: Overview of a few of the most used distance metrics in data processing and their mathematical description. All representations calculate the distance d between data point x and y.

Correlation distance. They evaluated the performance using the Calinski-Harabasz (CH) index [82], and found that the Euclidean distance metric consistently achieved the highest cluster quality in terms of the CH index [83]. As mentioned before, Aggarwal, Hinneburg and Keim proved that a fractional distance metric provides better understanding from a theoretical and empirical perspective, and can significantly speed up clustering algorithms like k-means clustering [50].

The choice of distance metric is not always motivated. Artemenkov and Panov used the Cosine similarity as distance function for their dimensionality reduction technique called NCVis, without explanation or consideration of other metrics [69]. Also Campello et al. applied Euclidean and Cosine distance on different data sets with no further discussion [84]. Verbeeck et al. argues that the choice of distance metric depends on the underlying structure of the data [3].

Chapter 3

Methods

3-1 Uniform Manifold Approximation and Projection (UMAP)

UMAP is a form of NLDR optimizing its LD embedding similarly to t-SNE, using force-directed graphs. UMAP gained traction over the last years as it showed promising results for a broad range of fields. Furthermore, UMAP consist of a well founded mathematical theory in topology and the ability to scale up towards large real-world data sets [11]. UMAP builds further on previous work of Belkin and Niyogi on Laplacian eigenmaps [47].

The UMAP algorithm consist in general of two important steps: (i) compute a probabilistic (fuzzy) graphical representation of the local relationships in the data, and (ii) optimize the LD embedding using Cross-Entropy (CE) as cost function by iterating using Stochastic Gradient Descent (SGD) [11]. These two steps have been visualized in Figure 3-1. In this subsection, both steps will be discussed separately considering their mathematical argumentation which the t-SNE is somewhat lacking [10].

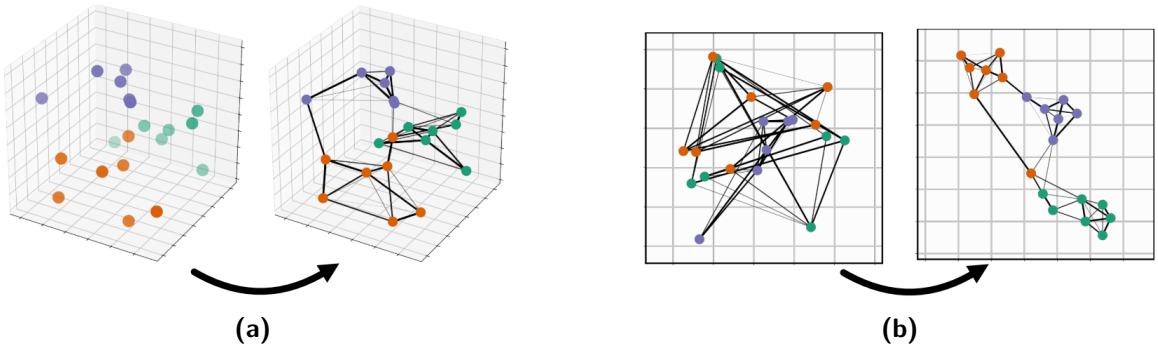


Figure 3-1: Overview of the two general steps of UMAP. **(a)** Computing the kNN graph based on the high-dimensional data and **(b)** optimizing the LD embedding that preserves the structure of the original graph [85]. The initialization determines the initial positions of the data points in low-dimensional space in the second step.

Step 1: Constructing the fuzzy graph In order to construct the fuzzy graph, the main assumption that UMAP makes, is that data is uniformly distributed on the manifold [11]. This is rarely the case with real-world data, but allowed when the manifold has a Riemannian metric not inherited from the ambient space, then a metric can be found such that the data is approximately uniformly distributed [11]. Therefore, the remaining assumption is that the data is locally connected. The original paper mentions that the topological information can be obtained by a translation of each metric space in the family into a fuzzy simplicial set. By taking a fuzzy union across the entire family, all the incompatibilities of the different families (different metrics used across the data set) can be smoothed out. What this means in practice is that each data point has a fixed number of neighbors for the construction of the k-Nearest Neighbor (kNN)-graph, resulting in some data point having larger lengths between their neighbors than other data points. At a local level, the distances between nearest neighbors is thus different throughout the entire data set. "Taking the fuzzy union" means that we define far away neighbors of one data point be as far away as a data point having much closer neighbors. This is essentially the basics of the abbreviation UMAP. The final result is a fuzzy simplicial set that describes the underlying structure of manifold $\mathcal{M}$ [11].

Intermezzo: k-Nearest Neighbor (kNN) graph In order to construct this weighted k-Nearest Neighbor (kNN) graph, UMAP uses an algorithm developed by Dong et al., who proposed an efficient construction of kNN graphs by applying nearest neighbour (NN)-descent [86]. This approximates a kNN graph almost perfectly with only a few iterations, and uses the intuition that "my neighbours' neighbours are likely to be my neighbours" [68]. Computing the exact kNN graph has an complexity of $O(N^2 d)$ with N the number of data points and d the number of dimensions. This computation is often too costly, which led to approximations using 3 different approaches [68]: (i) space-partitioning trees, (ii) locality sensitivity hashing techniques and (iii) neighbor exploring techniques. Space-partitioning divides the space into different regions, organizing these regions can be done using different trees: k-d trees [87, 88], vantage point (vp) trees (also being used by t-SNE) [89], cover trees [90] and Random Projection (RP) trees [91]. UMAP uses the RP trees for construction of its space-partitioning tree, which have proven to be state-of-the-art in constructing kNN graphs [68]. Dasgupta and Freund report an empirical complexity for the NN-descent of $O(N^{1.14})$ [91]. The locality sensitivity hashing techniques are being used to map data points into different buckets and data points in the same buckets are likely to be similar to each other [68]. NN-descent falls in the category of neighbour exploring techniques.

The probability distribution in high-dimensional space is exponentially defined as follows:

$$p_{i|j} = \exp\left(\frac{-\max\left(0, d\left(x_i, x_j\right) - \rho_i\right)}{\sigma_i}\right), \quad \rho_i = \min\left\{d\left(x_i, x_{i_j}\right) \mid 1 \leq j \leq k, d\left(x_i, x_{i_j}\right) > 0\right\}, \tag{3-1}$$

with $d(x_i, x_j)$ the distance metric between x_i and x_j in high-dimensional space and k the number of nearest neighbours, which is one of the important hyperparameters of UMAP. Parameter σ ensures a smooth approximation of the kNN-distance, and will be tuned such that the following relationship is met:

$$k = 2^{\sum_{j=1}^{k} p_{i|j}} \tag{3-2}$$

A weighted directed graph $G = (V, E, w)$ can be constructed with the vertices $V = X$ (X being a matrix containing distances between k nearest neighbours of X_i, and it holds that $X_i \in X$), the directed edges are formed by $E = \left\{\left(x_i, x_{i_j}\right) \mid 1 \leq j \leq k, 1 \leq i \leq N\right\}$ and the weight w are equal to the probabilities in the high-dimensional space $p_{i|j}$ [11]. As mentioned before, by taking the fuzzy union across the entire family, incompatibilities can be smoothed out. Symmetrization aims to achieve this by gluing together all locally different distance metrics, which can be written mathematically as follows:

$$p_{ij} = p_{i|j} + p_{j|i} - p_{i|j} \circ p_{j|i} \tag{3-3}$$

An important difference in the symmetrization step of UMAP compared to t-SNE is that t-SNE normalizes over all data points at this step, which is assumed to be a computational drawback and results in a slower run time [11]. The resulting weighted kNN graph G will be optimized in the next step.

Step 2: Optimizing the LD embedding The probabilities in the LD embedding are given by a family of curves which can be seen in Equation 3-4 [11]. This representation is virtually identical to the Student t-distribution of t-SNE.

$$q_{ij} = \left(1 + a\left(y_i - y_j\right)^{2b}\right)^{-1} \tag{3-4}$$

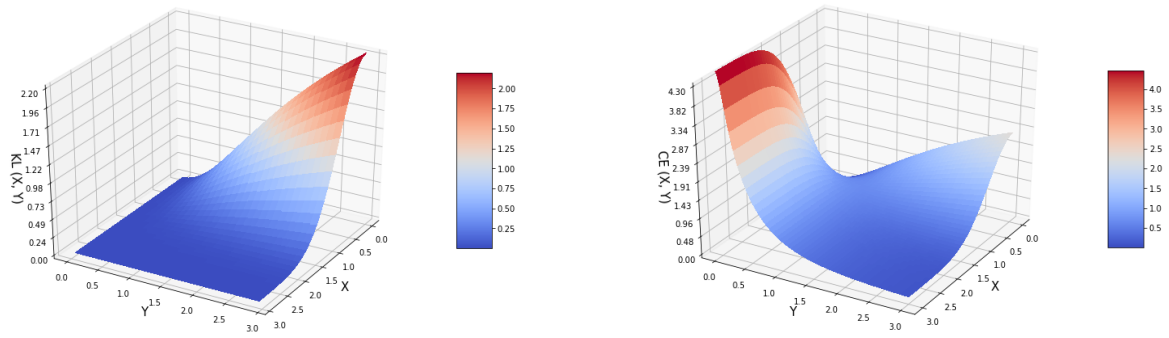
Parameters a and b are found through nonlinear least-square fitting with the use of hyperparameter `min_dist`.

$$\left(1 + a\left(y_i - y_j\right)^{2b}\right)^{-1} \approx \begin{cases} 1 & \text{if } y_i - y_j \leq \texttt{min\_dist} \\ e^{-(y_i - y_j) - \min\_\text{ dist}} & \text{if } y_i - y_j > \texttt{min\_dist} \end{cases} \tag{3-5}$$

Where t-SNE uses the KLD as cost function, UMAP is minimizing the CE. This CE can be described by Equation 3-6 [11]. Similar to the KLD, this equation can be seen as a spring system in which the first part of the equation is describing the attractive force and the second part the repulsive force. Using the approximate representations of the probabilities in the low and high dimension $P(X) \approx e^{-X^2}, \quad Q(Y) \approx \frac{1}{1+Y^2}$, the CE can be rewritten in terms of only the distances (assuming X is the distance between data points in the high-dimensional space and Y in the LD embedding, and ignoring the constant terms with only $P(X)$ [73]).

$$\begin{aligned} CE(X, Y) &= \sum_i \sum_j \left[p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) + (1 - p_{ij}) \log\left(\frac{1 - p_{ij}}{1 - q_{ij}}\right) \right] \\ &= P(X) \log\left(\frac{P(X)}{Q(Y)}\right) + (1 - P(X)) \log\left(\frac{1 - P(X)}{1 - Q(Y)}\right) \\ &= e^{-X^2} \log\left[e^{-X^2}\left(1 + Y^2\right)\right] + \left(1 - e^{-X^2}\right) \log\left[\frac{\left(1 - e^{-X^2}\right)\left(1 + Y^2\right)}{Y^2}\right] \\ &\approx e^{-X^2} \log\left(1 + Y^2\right) + \left(1 - e^{-X^2}\right) \log\left(\frac{1 + Y^2}{Y^2}\right) \end{aligned} \tag{3-6}$$

The resulting penalization that is being imposed by the optimization algorithm has been visualized by Oskolkov, giving an interpretation of the huge differences in the KLD of t-SNE and CE of UMAP [73]. To this end, the visualization of the cost functions of t-SNE and UMAP (Equation 2-16 and Equation 3-6, respectively) can be seen in Figure 3-2. For t-SNE, it is shown in Figure 3-2a that when the distances in high-dimensional space X are small and the distances in low-dimensional space Y are large, the forces imposed by the optimization algorithm are stronger (local structure). There where the cost function is minimal (the ground plane), it can be seen that the LD embedding is a near-perfect representation of the high-dimensional data. Here, the distances in high and low-dimensional space are equal to each other ($Y = X$), so no optimization will be conducted for those data points anymore. Therefore, it can be seen in Equation 2-18 that when X is large, no optimization is happening since it sees no difference in distances in Y. This describes the global structure which the t-SNE algorithm is not using in its optimization step at all. A clear difference by UMAP can be seen in Figure 3-2b, which also penalizes the LD embedding when the distances in Y are small and in X are large. It is visualised that UMAP is better at preserving global structure as opposed to t-SNE [73].



(a) **(b)**

Figure 3-2: Comparison of the cost function **(a)** Kullback–Leibler divergence (KLD) and **(b)** Cross-Entropy (CE) that are used in t-SNE and in UMAP, respectively. Where both algorithms preserve local structure, it can be seen that UMAP preserves the global structure of the data better with the use of CE due to the increasing penalization when distances in the high-dimensional data X are big and distances in the LD embedding Y are small. The penalization value on the z-axis is arbitrary [73].

Intermezzo: Stochastic Gradient Descent (SGD) and Negative sampling (NEG) In order to optimize the LD embedding, UMAP uses SGD with NEG [92] as opposed to the gradient descent of t-SNE. This combination allows the LD embedding of UMAP to be of arbitrary dimension K, unlike t-SNE which works only for two or three dimensions due to increasing computational complexity with K [11]. NEG has previously also been applied by LargeVis [68]. The definition of NEG by Mikolov et al. can be seen in Equation 3-7 [92]. A thorough explanation and derivation of this expression can be found in a study by Goldberg and Levy [93].

$$\log \sigma \left({v'_{w_O}}^{\top} v_{w_I} \right) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma \left(-{v'_{w_i}}^{\top} v_{w_I} \right) \right] \tag{3-7}$$

This equation uses the sigmoid function $\sigma(x) = \frac{1}{1+\exp -x}$. The first and second part of the equation respectively models the observed and negative edges drawn from the noise distribution, with k the number of negative samples [70].

SGD can be parallelised, which could possible result in a decrease of computational time for UMAP in the future [11]. This form is generally known as asynchronous SGD, and has proven to be very efficient and effective in sparse graphs [94], and is useful for us since `n_neighbors` is much smaller than the number of pixels N. Tang et al. discusses the implementation of SGD for LargeVis as well, which is another NLDR technique [68]. Instead, they adopt an approach called edge sampling from their previous work, in order to not influence the objective function and learning process of LargeVis [70].

SGD requires the gradient in order to optimize the cost function, similarly to GD. The gradient of the cost function of UMAP can be obtained by differentiating CE in Equation 3-6 and by filling in the expression of the LD embedding of Equation 3-5. The resulting gradient of the CE can be seen in Equation 3-8. In practice, UMAP uses a force directed graph layout. This LD embedding will be optimized by applying forces along the edges of the graph and between randomly sampled nodes of the graph, described as attractive and repulsive forces, respectively [11].

$$\frac{\delta CE}{\delta \mathbf{y_j}} = \left(\underbrace{w\left((x_i, x_j)\right) \frac{-2ab \left\|\mathbf{y_i} - \mathbf{y_j}\right\|_2^{2(b-1)}}{1 + \left\|\mathbf{y_i} - \mathbf{y_j}\right\|_2^2}}_{Attractive force} + \underbrace{\left(1 - w\left((x_i, x_j)\right)\right) \frac{b}{\left(\epsilon + \left\|\mathbf{y_i} - \mathbf{y_j}\right\|_2^2\right)\left(1 + \left\|\mathbf{y_i} - \mathbf{y_j}\right\|_2^2\right)}}_{Repulsive force} \right) (\mathbf{y_i} - \mathbf{y_j}) \tag{3-8}$$

In this equation, $w((x_i, x_j))$ are the probabilities between data points i and j of the high dimensional space and ϵ is a small constant added to the equation in order to avoid division by zero [11]. A clear distinction between the attractive and repulsive forces can be seen. Parameters a and b are the nonlinear fitted parameters determined by the hyperparameter `min_dist`, and the default settings of UMAP are $a \approx 1.929$ and $b \approx 0.7915$. With values $a = b = 1$, the distribution would result in the Student t-distribution used in t-SNE [11].

Alternative versions of UMAP While UMAP is relatively new, a few alterations have already been developed. Nolet et al. have successfully obtained speedups of up to 100x in practice, by introducing cuML UMAP [95]. They found that a large part of the computation was constructing the kNN graph (26%), therefore cuML UMAP only computes the kNN graph once. Nolet et al. also mention that more improvements can be made in adjusting the hyperparameters correctly, although the original paper only applied a couple different hyperparameter configurations [95].

Sainburg et al. proposed Parametric UMAP, which replaces the step of optimizing the embedding with SGD with a deep neural network that learns a parametric relationship between the high-dimensional space and LD embedding [85]. However, parameteric UMAP still uses the default parameters of UMAP when starting the algorithm. They concluded that the performances of the parametric embedding are similar to the non-parametric embedding, except the added advantage of learned mapping by parametric UMAP [85]. However, there are still parameters left for the user to adjust, which may require supervision by an expert. In general, Parametric UMAP needs more time to achieve the same level of quality compared to regular UMAP.

Existing approaches up until now often neglect the local density of the data. This may lead to misleading interpretations and possibly missing underlying biological structure. Den-SNE and densMAP aim to take the local density of the data into account by adding a term to the cost functions, as can be seen in Equation 3-9 [56].

$$\begin{aligned}
\mathcal{L}^{\text{den-SNE}} &= \text{KLD}\left(P^{\text{t-SNE}} \| Q^{\text{t-SNE}}\right) - \lambda \,\text{Corr}\left(r_o^{\text{t-SNE}}, r_e^{\text{t-SNE}}\right), \\
\mathcal{L}^{\text{densMAP}} &= \text{CE}\left(P^{\text{UMAP}} \| Q^{\text{UMAP}}\right) - \lambda \,\text{Corr}\left(r_o^{\text{UMAP}}, r_e^{\text{UMAP}}\right), \\
\text{With the correlation coefficient: } &\text{Corr}\left(r_e, r_o\right) = \frac{\text{Cov}\left(r_e, r_o\right)}{\left(\text{Var}\left(r_e\right)\text{Var}\left(r_o\right)\right)^{1/2}}
\end{aligned} \tag{3-9}$$

Here, λ is a hyperparameter tunable by the user, based on how strongly the density-preservation term should operate compared to the original cost function. Narayan et al. define $r_o\left(x_i\right) := \log R_o\left(x_i\right)$ and $r_e\left(y_i\right) := \log R_e\left(y_i\right)$, with R_o and R_e the radii respectively in the high and low-dimensional space [56]. The additional information that was captured gave more biological insight into immune cell transcriptomic variability in tumors, specialization of monocytes and dendritic cells and temporally modulated transcriptomic variability across developmental lineages of *C. elegans* [56]. Narayan et al. report that den-SNE and densMAP run approximately 20% slower than their original counterpart. Only one hyperparameter configuration was used, with the Euclidean distance metric.

3-1-1 Initialization phase

The initialization phase of the applied dimensionality reduction technique remains important w.r.t. how fast a global optimum will be found, and whether this global optimum will be found at all. Independent Component Analysis (ICA), a linear decomposition technique, uses an iterative approach to converge towards a global optimum. Multiple runs, random initialization and combining the results over these different runs are often applied to avoid getting stuck in a local minimum [3].

for the same reason, t-SNE is initialized by samples drawn from a isotropic Gaussian distribution with a small variance centered around the origin added [6]. Additionally, t-SNE also adds a large momentum term α to its gradient, which will decay throughout the iterations. This not only avoids poor local minima, but also speeds up the optimization [6]. This practice is also referred to as early exaggeration, and is proposed as a fixed factor of 4 in the original algorithm of t-SNE (also visible in Equation 2-21). The early exaggeration ensures that data points are spread out farther away at the start of the algorithm, allowing more space for the data points to converge towards their global optimum. Linderman and Steinerberger point out that the early exaggeration phase of t-SNE can be improved [10] since the original paper on t-SNE of Van der Maaten provides no proof for the selection of this parameter [6]. They also point out that it can be empirically checked that a higher value than one for the early exaggeration phase improves the LD embedding. Belkina et al. attempted to calibrate the early exaggeration by utilizing the KLD in real time in a data-driven way [12]. Consequently, the resulting embedding achieved a lower value for the KLD as opposed to the default settings of t-SNE, showing the importance of the correct parameter configuration for different data sets.

NCVis employs the Power Iteration method during the initialization phase [69]. This method ensures a more sophisticated approximation of the positions of the data points in the LD embedding, by already taking the kNN-graph into account. This kNN-graph will be calculated prior to the initialization.

UMAP also incorporates information from the kNN-graph, and assigns initial low-dimensional coordinates through the Graph Laplacian [11]. The initial kNN-graph will be constructed using matrix factorization, formally called the normalized Laplacian matrix [47].

$$L = D^{1/2}(D - A)D^{1/2}, \tag{3-10}$$

where A is the adjacency matrix and D the degree matrix of graph G. Kobak and Linderman stress that the main difference between t-SNE and UMAP lies within this initialization phase, arguing that both algorithms would preserve global structure poorly if UMAP would be initialized similar to t-SNE (random initialization) [96]. In previous work, Kobak claimed that FIt-SNE is as quick as UMAP, and even up to four times quicker when applied to two different data sets (while both algorithms are configured with default settings) [97]. This claim is contradictory to findings of McInnes et al., who showed that UMAP outperformed FIt-SNE in terms of runtime for every subsample of the full Google News data set [11].

Spatial initialization Besides the spectral and random initializations within UMAP, a third initialization is proposed: the spatial initialization. This initialization utilizes the original pixel locations as they are registered during the ionization of the tissue surface, and uses these coordinates as initial coordinates of data points in the LD embedding. This means that the LD embedding is being initialized in a 2-D plane due to the flat surface nature of most IMS data sets. Therefore, the embedding cannot exist in a dimension higher than $K = 2$, since the forces applied to the constructed kNN graph can only exist in the 2 dimensions and any other dimension would be orthogonal to the first two dimensions. The spatial initialization is thus dependent on the acquisition-dimension. With 3-D IMS data sets [20], this initialization can be used as starting positions for data points in a 3-D LD embedding. Further investigation of

this initialization and a comparison with the standard procedures of UMAP will be conducted in a separate experiment, formulated in subsection 4-2-2.

3-1-2 Applications of UMAP

The fast algorithm, scalability to large data sets, mathematically well grounded theory and interpretable visualizations make UMAP a good candidate for analyzing IMS data. Smets et al. stress the fact that UMAP is able to capture all features irrespectively of the number of dimensions K spanning the LD embedding, which is important when visualizing the data since the underlying manifold may not be restricted to 2-D or 3-D [98]. They applied a bidirectional dimensionality reduction approach by taking both the spatial and spectral information into account. Based on this information, the m/z bins were ranked by importance and the results produced by UMAP were confirmed using the spatial colocalization of the identified ions. Results are likely to have biological meaning, allowing a better understanding of the underlying biological activities [98].

When used in identification of protein-protein interactions, UMAP shows higher sensitivity compared to traditional dimensionality reduction methods [99]. Dorrity et al. also mention the ability of capturing local as well as global structure, which makes UMAP a valuable addition to the identification of protein complexes, pathways, and novel interactions in transcriptomic data sets [99].

While t-SNE and UMAP may not always be able to segregate data points into complete clusters, at least they both surpass PCA in terms of identifying the underlying manifold [100]. In the same study, it was found that UMAP was able to extract the differentiation stage of T-cells within each major cluster, while t-SNE was unable to make them easily identifiable due to the absence of an perceptible structure [100]. In terms of speed, UMAP produces the LD embedding the fastest, comparable with FIt-Stochastic Neighbor Embedding (SNE). Becht et al. conclude that UMAP appears to be a robust method for dimensionality reduction, capable of preserving both local and global structures [100].

As mentioned in section 2-3, Cao et al. investigated the transcriptional dynamics of mouse organogenesis with single-cell RNA sequencing (scRNA-seq), then applied UMAP for constructing the LD embedding and identified hundreds of cell types and 56 trajectories of 61 embryos [80]. With this global view of mammalian organogenesis, it was shown that dynamical biological processes and trajectories could be identified. A detailed analysis of the developmental trajectories of cells has been conducted by Packer et al., which also looked at global scales [101]. A 3-D LD embedding obtained by UMAP was able to reconstruct the developmental trajectories, spanning the timeline from mid-gastrulation to terminal differentiation [101].

Bagger et al. applied a clustering technique called k-means clustering in a 10-D UMAP LD embedding, in order to cluster unlabeled single cells and color-code the obtained clusters [102]. The goal was to improve Bloodspot, a gene-centric database of mRNA expression of haematopoietic cells. In an effort of combining multiple scRNA-seq data sets, Park et al. developed batch balanced k-nearest neighbours (BBKNN) which succesfully connects cell populations across multiple data sets [103]. UMAP is used to convert the distances in the produced kNN graph by BBKNN to connectivities (for visualisation) [103].

3-2 Quality control for Low-Dimensional (LD) embedding

Evaluating a produced embedding by any dimensionality reduction technique remains a difficult objective, and often requires supervised inspection by experts afterwards [104]. This section will dive more into different methods of comparing a LD embedding to the high-dimensional data.

3-2-1 Spectral comparison

During a spectral comparison, the spectral information of an IMS data set is evaluated against the produced embedding. Effectively, the spectral information is utilized by UMAP in order to construct a low-dimensional feature space by variables that are a nonlinear combination of the original features.

3-2-1-1 Cross-Entropy (CE)

The Cross-Entropy (CE) is the measure that is minimized by UMAP following the procedure described in section 3-1. Not the full CE is being computed during optimization of the embedding, but shortcuts are made to compute a subset of the forces applied to the graph. The attractive forces are based on `n_neighbors` of the kNN-graph, and the repulsive forces are controlled by the negative sampling rate (only a fraction of the repulsive forces will be computed at each epoch). The forces applied by UMAP are effectively the derivatives of the CE, see Equation 3-8. To compute the full CE for evaluation, the following equation needs to be solved:

$$\begin{aligned} CE &= P\log\left(\frac{P}{Q}\right) + (1-P)\log\left(\frac{1-P}{1-Q}\right) \\ CE &= \underbrace{P\log(P)}_{constant} - P\log(Q) + \underbrace{(1-P)\log(1-P)}_{constant} - (1-P)\log(1-Q) \\ CE &= -(P\log(Q) + (1-P)\log(1-Q)), \end{aligned} \tag{3-11}$$

with P and Q the probabilities in high and low-dimensional space, respectively. Any constants are removed from the equation, since we are interested in the minimum of the CE. From here, two versions of the CE can be computed: the approximate or exact version, whose equations can be seen in Table 3-1. Note that the approximate equations for A and Q are the exact formulations with $a = 1$, $b = 1$, $\rho = 0$ and $\sigma = 1$. The approximation helps us visualize the basic principles of the behaviour of the cost function, as the CE has been compared to the Kullback–Leibler divergence (KLD) in Figure 3-2.

Somewhat underexposed in the original paper of UMAP, is the *gluing back together* property of UMAP for the high-dimensional probabilities. The algorithm allows interpolation between Union and Intersection of the high-dimensional probabilities with the parameter `set_op_mix_ratio` with values of 1.0 (default) and 0.0, respectively. This symmetrization is necessary to deal with the locally varying metrics found through parameter ρ. As a result, an undirected weighted graph is constructed such that the weights between each data point

	CE approx	CE exact
$A =$	$\exp\left(-X^2\right)$	$\exp\left(\frac{\max(0,X^2-\rho)}{\sigma}\right)$
$Q =$	$\frac{1}{1+Y^2}$	$\frac{1}{1+aY^{2b}}$

Table 3-1: Approximated and exact computation of the high and low-probability matrices A and Q. Here, X and Y are the distances between the data points in high and low-dimensional space, respectively. a, b, ρ and σ are parameters computed by UMAP. a and b are depended on the UMAP parameters `min_dist` and `spread`. ρ and σ are depended on the used metric in high-dimensional space and the UMAP parameter `n_neighbors`.

i and j are similar to weights between each data point j and i. The corresponding equation is as follows:

$$\begin{aligned} P &= \texttt{set\_op\_mix\_ratio}(A + A^\top - A \circ A^\top) + (1 - \texttt{set\_op\_mix\_ratio})A \circ A^\top \\ P &= \texttt{set\_op\_mix\_ratio}(A + A^\top - 2A \circ A^\top) + A \circ A^\top \end{aligned} \tag{3-12}$$

$$\texttt{set\_op\_mix\_ratio} = 1.0 \xrightarrow{Union} P = A + A^\top - A \circ A^\top$$
$$\texttt{set\_op\_mix\_ratio} = 0.0 \xrightarrow{Intersection} P = A \circ A^\top$$

with $\circ$ the Hadamard (or pointwise) product [11]. The exact computation of the CE takes the fuzzy topological representation of the data set into account [11]. Thus, for evaluation, similar parameters should be considered for a fair comparison. Throughout this thesis, when the CE is computed over the spectral domain, we mean the exact computation of CE.

3-2-1-2 Graph comparison

UMAP is constructing a graph in the high and low-dimensional space to discover relationships between data points and optimize the resulting embedding, respectively. This is also referred to as a graph embedding, which simply denotes the transformation of the high-dimensional data to a set of vectors. The two graphs can be compared to each other in an unsupervised way, as has been demonstrated by Kamiński et al. [104]. A *divergence score* is proposed to estimate how similar an embedding is compared to the original graph. The Jensen–Shannon divergence (JSD) is being used to compare the number of edges within and outside clusters in the data set. The JSD can be seen as a smoothed version of the KLD [104]. However, in order to run unsupervised, the algorithm requires a stable graph clustering algorithm to classify whether multiple data points are similar to each other, and subsequently determine which edges lay fully within the same cluster and which edges do not. Clustering becomes problematic for high-dimensional spaces such as real-world IMS data sets due to the curse of dimensionality, as has been discussed in subsubsection 2-2-1-2.

3-2-2 Spatial information

Besides the available chemical information of an IMS data set, the pixel coordinates are also known. Slicing the 3-D datacube at a specific m/z value produces an ion image such as in

Figure 2-9b. Spatial properties of data is used by algorithms like Maximum Autocorrelation Factorization (MAF) to reduce noise and extract cleaner signals [54]. MAF does this by maximizing the spatial autocorrelation of neighbouring pixels (range = 1 pixel). It is found that close-by measurement spots on IMS data sets show dependence on neighbouring pixels [105]. Since the dependency is based upon the pixel coordinates, this phenomenon is also referred to as spatial autocorrelation. Smets et al. also expected a certain degree of spatial autocorrelation, due to the spatial nature of IMS data, hence they applied a spatial-autocorrelation function to evaluate obtained embeddings [78]. This subsection will discuss further two procedures to take spatial information into account: spatial autocorrelation and constructing a spatial graph.

3-2-2-1 Spatial autocorrelation

Spatial autocorrelation describes the level of correlation between a single variable and variables positioned spatially close on a 2-D surface, resulting in data being not independent but variables being tied together [106]. In IMS, the variables in this context are the pixels of the image, which would mean that pixels influence their neighbouring pixels. Positive values for the spatial autocorrelation would indicate in relatively large clusters in the image domain, whereas negative values suggest that the pixels are dispersed. Spatial autocorrelation equal to zero would mean that the pixels' values are randomly distributed. This effect of spatial autocorrelation in IMS has been evaluated by Cassese et al. [105]. The presence of autocorrelation was assessed using Moran's I [107, 108], which is a popular measurement for global spatial autocorrelation due to its simplicity and fast runtime and will be separately discussed in this section.

Global measure: Moran's I Moran's I takes the values of neighboring pixels into account to determine whether the spatial structure is either negatively or positively spatially autocorrelated, or if the pattern is randomly distributed. Mathematically, the Moran's I statistic can be described using Equation 3-13 [107].

$$I = \frac{N}{W} \frac{\sum_i \sum_j w_{ij} \left(x_i - \bar{x}\right) \left(x_j - \bar{x}\right)}{\sum_i \left(x_i - \bar{x}\right)^2}, \tag{3-13}$$

with N the number of pixels, x the expression value of the pixel of interest, $\bar{x}$ the mean expression value for the close-by pixels, w_{ij} a matrix of weights describing how far away a neighbouring pixel is (zeros on the diagonal) and W the sum of all w_{ij} [80]. As can also be seen in Figure 3-3, values for Moran's I lie within $[-1, 1]$; with $I = -1$ being totally disperse, $I = 0$ denoting a random pattern and $I = 1$ showing a maximization of the spatial autocorrelation. Cassese et al. found that IMS data does indeed suffer from autocorrelation, and that this would only increase with decreasing pixel size. A Gaussian Conditional Autoregressive (CAR) model was used successfully to correct for spatial autocorrelation in IMS data [105]. Spatial autocorrelation has also been used by Cao et al. in order to identify genes with complex trajectory-dependent expression [80].

While not solely aiming for a maximization of the spatial autocorrelation, the Moran's I can aid the spectral evaluation to find the optimum embedding. Here, the we utilize the findings

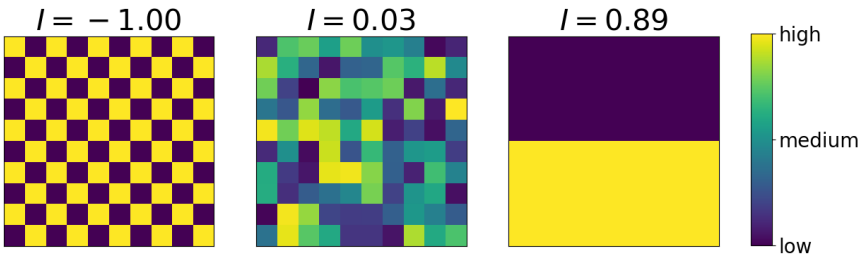


Figure 3-3: Three examples of spatial autocorrelation and the corresponding Moran's I value. Left shows a disperse image with $I \approx -1$. The middle image denotes a random pattern, hence $I \approx 0$. Finally, on the right the spatial autocorelation is maximized with 2 big clusters and $I \approx 1$.

that IMS is spatial autocorrelated and the spatial information that is being accompanied and yet to be implemented in Nonlinear Dimensionality Reduction (NLDR) techniques.

Local measure: Pearson correlation coefficient In order to compare different produced embeddings to each other, Smets et al. diagonally shifted the embedding over a number of pixels and computed the Pearson correlation coefficient between the started section and the shifted section [78]. When both sections are identical, the coefficient would show $R = 1$, where completely opposites would result in $R = -1$. The equation for the Pearson correlation coefficient is as follows:

$$R = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X) \cdot \text{Var}(Y)}}, \tag{3-14}$$

with X and Y the compared embeddings, Cov() and Var() the covariance and variance functions, respectively. While this measure is not ideal to compare high-dimensional data with the LD embedding, it is possible to compare the spatial structure between multiple embeddings as shown by Smets et al [78].

3-2-2-2 Compute CE using spatial information

Moving away from spatial autocorrelation, it is also possible to consider the spatial information of IMS data similar to the spectral information. Similarly to the construction of a graph in the high-dimensional spectral data, a graph can be constructed between neighboring pixels in the image domain. Figure 3-4 shows the neighboring pixels for varying thresholds around one pixel. The Euclidean distance metric is being used to compute the spatial similarity between pixels and construct the spatial kNN graph $\mathcal{G}_{\text{spatial}}$.

Similar to how UMAP finds it parameters for the computation of the CE using spectral data, these parameters can also be found following the same procedure for the spatial evaluation. With $\rho_{spatial}$, $\sigma_{spatial}$, $a_{spatial}$ and $b_{spatial}$ the CE can be computed comparably to the derivation in subsubsection 3-2-1-1. Since the pixels are uniformly spread across the "HD data" (e.g.

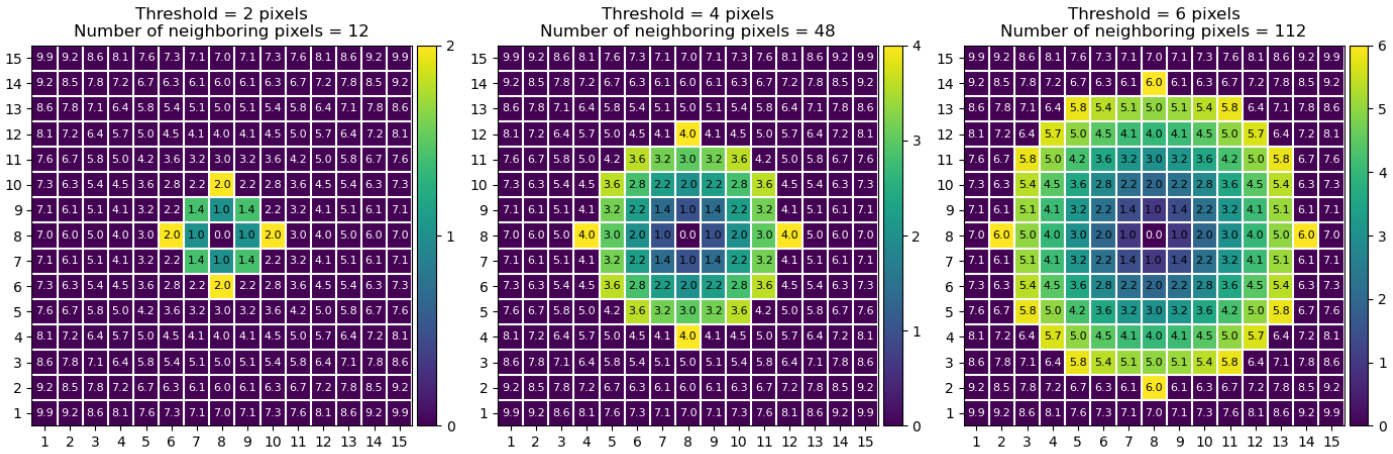


Figure 3-4: Comparison of the number of neighbors for varying thresholds (2, 4 and 6 pixels) using the Euclidean distance metric. The distance between each pixel i and pixel $(8, 8)$ is printed within each pixel. The number of neighboring pixels grows exponential with increasing threshold.

2-D grid), the approximate computation of $A_{spatial}$ and $Q_{spatial}$ as described in Table 3-1 is sufficient and would only result in a scaled version of CE_{exact}.

Where a minimization of the CE using spectral information would mean that pixels are similar in chemical content, one could conclude that a low value of the CE using spatial information indicates a preservation of the pixel's relative locations. Neighboring pixels with relatively large distance between each other are being penalized and will result into a higher value for the CE. From now on, the CE using spectral and spatial information will be referred to as $\text{CE}_{\text{spectral}}$ and $\text{CE}_{\text{spatial}}$, respectively.

3-2-3 Combining spectral and spatial information

Evaluation of the produced embeddings uses mostly spectral information only, since the embeddings utilize this information to construct their low-dimensional feature space with variables that are (non)linear combinations of the original m/z bins. Scoring the embedding based on their spatial properties would result in neglecting the spectral information, and is thus not desirable. Possibly the evaluation phase of the embeddings can benefit from both spectral and spatial information, if it were feasible to combine both measures. Both $\text{CE}_{\text{spectral}}$ and $\text{CE}_{\text{spatial}}$ lie within $[0, 1]$, where $\text{CE} = 0$ implies the best possible representation of the original data. The spatial autocorrelation using Moran's I produces values within $[-1, 1]$. In order to combine the Moran's I with $\text{CE}_{\text{spectral}}$, the range is transformed to also fit within $[0, 1]$. Now, $I = 0$ indicates fully positive spatial autocorrelation and $I = 1$ fully negative spatial autocorrelation, since we are interested in finding a maximization of the spatial autocorrelation up to a certain degree. The transformation can be written as follows.

$$\text{CE}_{\text{spatial, Moran's I}} = \frac{-I + 1}{2} \tag{3-15}$$

Now, with both spectral and spatial scoring measures within the $[0, 1]$ range, they can be combined to evaluate an embedding produced by UMAP overall. While $\text{CE}_{\text{spatial, Moran's I}}$ is not exactly the measured cross-entropy, the notation is used to indicate the transformed Moran's I. Employing Moran's I as a scoring metric would effectively work as a "smoothing filter" across the produced embedding. The underlying spatial graph is very similar for $\text{CE}_{\text{spatial, graph}}$ and $\text{CE}_{\text{spatial, Moran's I}}$. In experiment 2, we investigate what form of spatial information should be used as complement for spectral information.

From here on, the spatial information will be referred to as $\text{CE}_{\text{spatial}}$. In order to interpolate between the importance of $\text{CE}_{\text{spectral}}$ and $\text{CE}_{\text{spatial}}$, the yet to be introduced parameter $\lambda_{spatial}$ which belongs to the UMAPLUS implementation (subsection 3-3-1), will be included in the scoring metric. In short, $\lambda_{spatial}$ controls the importance between using spectral and spatial information during the optimization of the LD embedding by UMAP. The importance of spectral information is leading and will always be greater or equal than the spatial information ($\text{CE}_{\text{spectral}} \geq \text{CE}_{\text{spatial}}$). This decision is made since we are more interested in chemical content than pixel locations. The intention is that the spatial information should merely help the spectral information in further improving the LD embedding. Mathematically, the evaluation function can be summarized in Equation 3-16.

$$\text{CE}_{combined} = \underbrace{\frac{1}{2}\left(1 + \lambda_{spatial}\right)\text{CE}_{\text{spectral}}}_{\text{Spectral component}} + \underbrace{\frac{1}{2}\left(1 - \lambda_{spatial}\right)\text{CE}_{\text{spatial}}}_{\text{Spatial component}} \tag{3-16}$$

The resulting loss function $\text{CE}_{combined} \in [0, 1]$ will be primarily used for embedding evaluation, having both spectral and spatial information combined. Visually, the effect of varying $\lambda_{spatial}$ on the fraction of the spectral and spatial components can be seen in Figure 3-5.

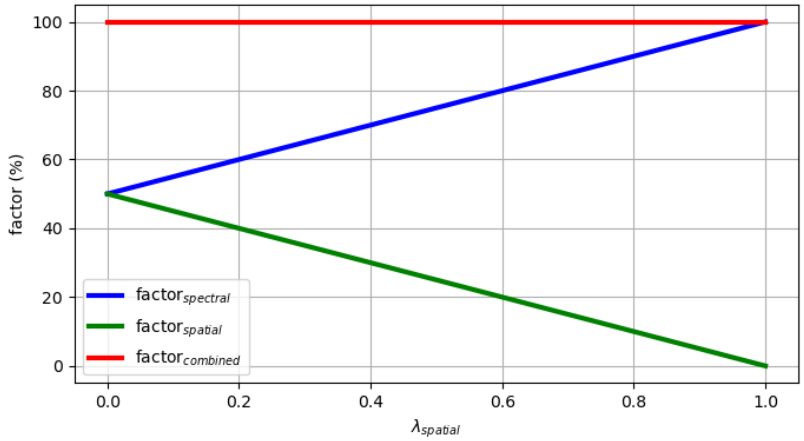


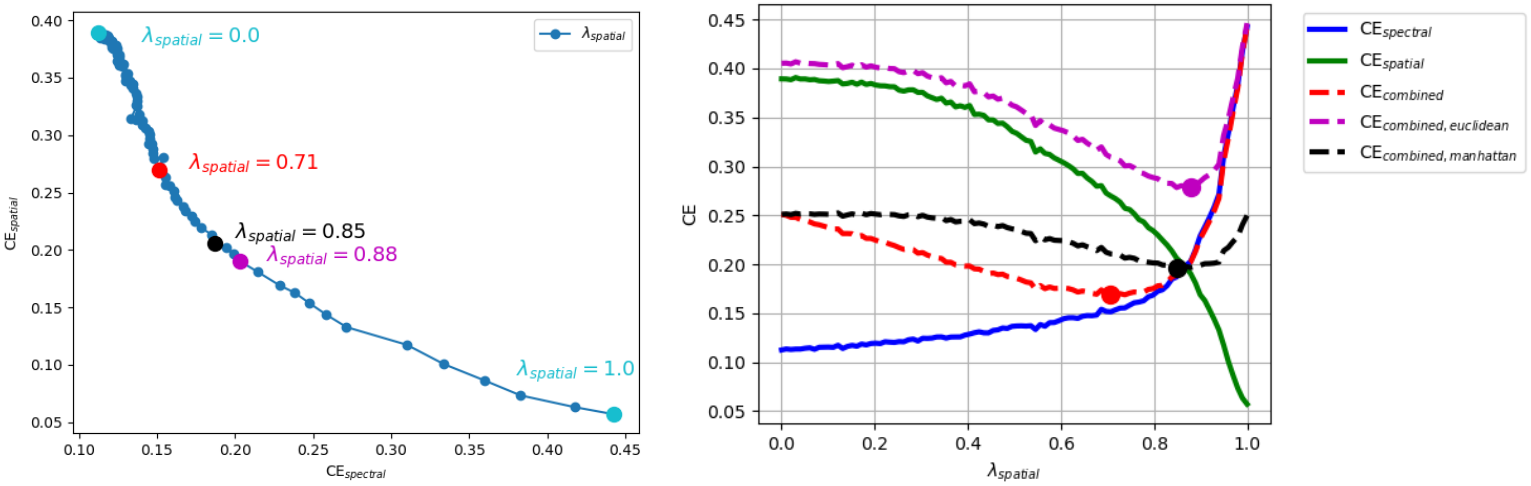
Figure 3-5: The fraction of both spectral and spatial components are visualized for varying $\lambda_{spatial}$. It can be seen that $\text{CE}_{\text{spectral}}$ will always be the most important scoring metric, and for increased $\lambda_{spatial}$ the importance of $\text{CE}_{\text{spectral}}$ grows whereas $\text{CE}_{\text{spatial}}$ shrinks towards zero for $\lambda_{spatial} = 1.0$

We have chosen for a notation of $\text{CE}_{combined}$ based on $\lambda_{spatial}$ in order to not give too much power to the spatial information during evaluation, since the spectral information is consid-

ered more valuable. Other equations for $\text{CE}_{combined}$ can be formulated. Let us consider the following formulas, combining $\text{CE}_{spectral}$ and $\text{CE}_{spatial}$ without the involvement of $\lambda_{spatial}$.

$$\begin{aligned} \text{CE}_{combined,euclidean} &= \sqrt{(\text{CE}_{spectral})^2 + (\text{CE}_{spatial})^2} \\ \text{CE}_{combined,manhattan} &= \frac{1}{2}\text{CE}_{spectral} + \frac{1}{2}\text{CE}_{spatial} \end{aligned} \tag{3-17}$$

The naming of these measures is based upon the usage of $\text{CE}_{spectral}$ and $\text{CE}_{spatial}$. It is effectively the distance from the origin in Figure 3-6a using the Euclidean and Manhattan distance metrics, respectively. Except, the Manhattan distance has been divided by 2 in order for $\text{CE}_{combined,manhattan}$ to stay within $[0, 1]$ (division does not matter for its optimum). The minimum value for $\lambda_{spatial}$ found by $\text{CE}_{combined}$, $\text{CE}_{combined,euclidean}$ and $\text{CE}_{combined,manhattan}$ can be seen in Figure 3-6, applied to the synthetic data set with additive Gaussian noise ($\sigma = 10$). Clearly visible is the lower value for $\lambda_{spatial}$ as minimum for $\text{CE}_{combined}$, indicating the fact of prioritizing $\text{CE}_{spectral}$ over $\text{CE}_{spatial}$. In subsection 4-2-4, we will investigate the effects of $\lambda_{spatial}$ more in depth.



(a) **(b)**

Figure 3-6: Comparison between the optimal value for $\lambda_{spatial}$ for different evaluation metrics on the synthetic data set with additive Gaussian noise ($\sigma = 10$). See subsection 4-1-1 for more information of the construction of these data set. **(a)** $\text{CE}_{spectral}$ is plotted against $\text{CE}_{spatial}$, and the minimums of the multiple evaluation measures are highlighted in their corresponding color. **(b)** $\lambda_{spatial}$ is varied between $[0.0, 1.0]$, and the embedding has been evaluated using multiple evaluation metrics.

Intermezzo: subsampling The distances in high and low-dimensional space X and Y respectively, of both the spectral and spatial information domain, contain all distances between data point i and j. These matrices are also known as the pairwise distance matrices of size $\mathbb{R}^{N \times N}$ with N the number of data points, and thus scale quadratically with respect to the number of data points. Common data types use floating point values float32 (4 bytes) or float64 (8 bytes), where calculations with variables in float64 notation can be much faster on 64-bits operating systems.

Assuming that a IMS data set contains 10^6 pixels, each represented by a float64 8 byte-precision number, the resulting pairwise distance matrices take $2 \cdot 64 \cdot 10^{6 \cdot 2} = 1.28 \cdot 10^{14}$bits = $1.6 \cdot 10^{13}$bytes = 16.0TB of available RAM on the workstation. The sheer size of this single matrix is often unfeasible on most workstations. As a solution, a subset of the total number of pixels can be taken to reduce the size of the pairwise distance matrices X and Y. The maximum number of data points possible for the evaluation of Equation 3-16 is based on the physical constraints of the workstation (e.g. memory for calculation) in float64 precision, and can be obtained using the following equation:

$$N_{max} = \sqrt{\frac{8 \cdot n_{bytes,max}}{2 \cdot 64}} = \sqrt{\frac{1}{16} n_{bytes,max}} = \frac{1}{4}\sqrt{n_{bytes,max}} \tag{3-18}$$

Here, the factor 8 leads from the conversion of bytes to bits, 64 denotes the float64 precision, the factor 2 comes from the fact that we both need X and Y matrices in memory and $n_{bytes,max}$ is the maximum allowable memory size for evaluation. For a maximum allowable memory size of 2.0 GB, only $N_{max} = 11,180 \approx 1.12 \cdot 10^4$ data points can be used. This is merely a fraction of a realistic estimate of 10^6 data points of a real-world IMS data set. Subsection 4-2-1 will explore the possibility of subsampling the data such that evaluation using Equation 3-16 is still possible.

3-3 Extensions of UMAP

This section will propose two extensions of UMAP: (i) UMAPLUS; the original UMAP algorithm with added optimization of the embedding using the spatial information of the IMS data set, and (ii) Data-Driven UMAP (DD-UMAP); using a correct evaluation method, DD-UMAP will assess the optimum parameter configuration of UMAP and thus produce the best possible LD embedding according to the evaluation function.

3-3-1 UMAP with integrated spatial information (UMAPLUS)

Besides evaluating the embedding using spatial information, we can use the pixel coordinates prior to dimensionality reduction and use this information during the optimization of UMAP. In order to do so, a spatial kNN-graph $\mathcal{G}_{\text{spatial}}$ is constructed in the 2-D image domain between pixels using the Euclidean distance metric. This graph is being optimized side-to-side with the kNN graph based on the spectral information $\mathcal{G}_{\text{spectral}}$. Here, we effectively force neighboring pixels to "stick" together under the assumption that IMS data is subjected to a certain degree of spatial correlation [78].

In order to interpolate between the importance of optimizing $\mathcal{G}_{\text{spectral}}$ and $\mathcal{G}_{\text{spatial}}$, the parameter $\lambda_{spatial} \in [0, 1]$ is introduced. This controls the fraction of forces applied on the data points based on either spectral or spatial information during optimization. Mathematically, the cost function of UMAPLUS will take the following form:

$$\begin{aligned}\mathcal{L}_{UMAPLUS}(X, Y) =& \lambda_{spatial} \cdot \mathrm{CE}\left(P_{spatial}\left(X_{spatial}\right), Q\left(Y\right)\right) + \\ & (1 - \lambda_{spatial}) \cdot \mathrm{CE}\left(P_{spectral}\left(X_{spectral}\right), Q\left(Y\right)\right)\end{aligned} \tag{3-19}$$

with $X_{spectral}$ and $X_{spatial}$ being the pairwise distance matrices using spectral and spatial information, respectively. Please note that $\lambda_{spatial} = 0$ causes only optimization of $\mathcal{G}_{spectral}$ and $\lambda_{spatial} = 1$ only optimization of $\mathcal{G}_{spatial}$. This is not to be confused with the usage of spectral and spatial information by the evaluation function in Equation 3-16. The default number of neighbors for $\mathcal{G}_{\text{spatial}}$ will be 12. This will corresponds to a threshold of 2 resulting in 12 neighboring pixels as described in Figure 3-4.

3-3-2 Data-Driven UMAP (DD-UMAP)

UMAP consist of many parameters that influence the produced embedding. Evaluating every parameter combination using grid search will cause an experiment to consist of $S = \prod_{k=1}^{K} \left|L^{(k)}\right|$ operations [109], with L the number of parameters and K the number of parameter evaluations. Cycling through every possible parameter configuration becomes unfeasible for the number of parameters coded within UMAP. Hence, a smarter search method would be beneficial. In this section, the parameters of UMAP are discussed, as well as optimization techniques to search through the parameter space, leading up to the creation of DD-UMAP.

3-3-2-1 UMAP parameters

This subsection sums up the most significant parameters. A complete list of parameters used by UMAP, with their default settings and short description, can be found in section A-1. The authors of UMAP and numerous studies have highlighted the following hyperparameters to be of greatest importance: `n_components`, `n_neighbors`, `min_dist`, `n_epochs` and `metric` [11]. In this subsection, these parameters and their effect on the LD embedding will be discussed more thoroughly.

`n_components` The number of components determines the dimension of the LD embedding K to which UMAP tries to reduce the number of features from the high-dimensional space M. For visualisation purposes, this will usually be set to 2 or 3. The underlying manifold could exist in an higher dimension, although visualizing such embeddings would become difficult. The number of components must be smaller than the original number of features in the high-dimensional data ($K < M$).

`n_neighbors` The number of neighbors determines the amount of neighbouring data points during the construction of the kNN-graph in the high-dimensional space. This parameter essentially balances the local versus global structure in the data. Low values of `n_neighbors` will force UMAP to search at a local scale for the underlying manifold. Large values result in more connections between data points in the kNN graph, and focus more on the global structure of the data. The choice for `n_neighbors` may depend on the intention of the user, and thus has a large impact on the interpretation of these results. Increasing `n_neighbors` increases the non-zero values in a kNN-graph, and for relative high values it can result in a long computational time of UMAP on large data sets.

`min_dist` The minimal distance controls the desired separation between data points which are positioned more closely in the LD embedding. Low values result in densely packed structures, but more likely to faithfully represent the underlying manifold. Larger values spread out the data more evenly, which helps for visualisations since it prevents potentially overplotting of data points. The authors of UMAP call this hyperparameter purely aesthetical [11], although it could matter for clustering algorithms since the position of the data points will be influenced.

`n_epochs` The number of epochs controls the number of training steps for optimizing the LD embedding during SGD. In the original UMAP implementation, 500 epochs is used for data sets with less than 10000 data points, and 200 epochs otherwise. Higher values result in more function evaluations and thus a "better" embedding. However, as can be seen in Figure 3-7, after a certain number of epochs, the LD embedding found its (global) optimum and further optimization is unlikely. Continuing with evaluating the cost function would be fruitless.

In order to produce a stable LD embedding, UMAP introduces the learning rate α, which guarantees that the forces reach zero within a predefined number of epochs. The learning rate α is essentially a multiplication of the forces applied by UMAP and decays linearly from 1.0 to 0.0 over the given number of epochs. The forces within UMAP will be reviewed in subsection 4-2-3.

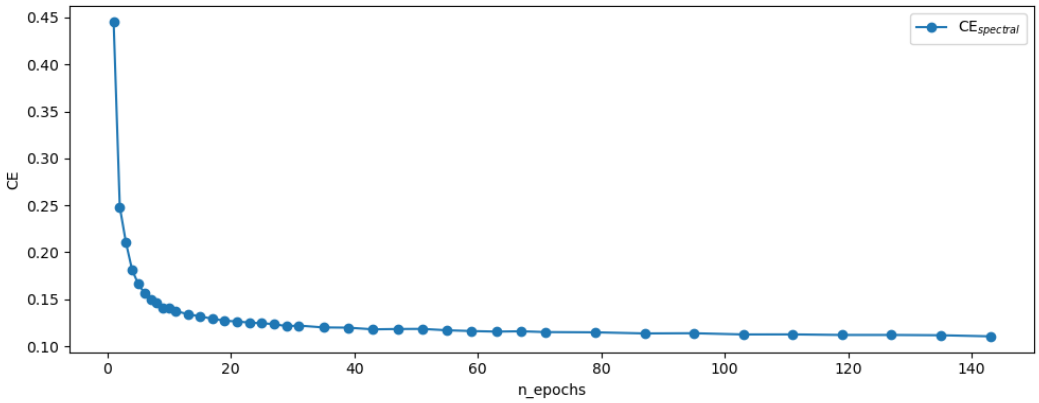


Figure 3-7: Evolution of the Cross-Entropy (CE) varying the number of epochs on the synthetic data set with additive Gaussian noise ($\sigma = 10$).

`metric` This parameter determines what distance metric is used in the high-dimensional embedding in the construction of the kNN-graph. The importance of the distance metric and previous research done on the effect of different distance metrics on IMS data have been discussed in section 2-3. UMAP has multiple distance metrics built in that can be viewed in the documentation of UMAP, but it also support custom distance metrics. Several distance metrics being used in literature are already discussed in section 2-3.

3-3-2-2 Premature parameter selection for optimization

Not all parameters make sense to optimize, such as Boolean parameters like `verbose` or `metric_kwds`, which is available for additional settings for the distance metric. A list of parameters this thesis will focus can be seen in Table 4-2, as well as their allowable range. Further experiments will check the sensitivity of the parameters. A final list of parameters to optimize will follow from subsection 4-2-3, accompanied with the necessary motivation.

Parameter	**Value range** $[min - max]$	**Data type**
`n_neighbors`	$[2 - 100]$	Integer
`learning_rate`	$[0.1 - 10.0]$	Float
`init`	[*"spectral"*, *"random"*, *"spatial"*]	Categorical
`min_dist`	$[0.0 - 1.0]$	Float
`spread`	$[1.0 - 3.0]$	Float
`set_op_mix_ratio`	$[0.1 - 1.0]$	Float
`repulsion_strength`	$[0.1 - 10.0]$	Float
`negative_sample_rate`	$[1 - 25]$	Integer
`spatial_lambda`	$[0.01 - 1.0]$	Float
`n_neighbors_spatial`	$[4, 12, 28, 48, 80, 112, 148, 196]$	Categorical

Table 3-2: List of most influenceable parameters within UMAP (included UMAPLUS) w.r.t. the produced LD embedding. The second column lists the allowable value range of the parameter. However for parameters of the type *Categorical*, this indicates the multiple allowable binary states of said parameter.

3-4 Optimization methods

3-4-1 Naive approaches

In this section, approaches that have been used for tuning the parameters of dimensionality reduction methods and any given algorithm will be discussed. For now, mainly two naive approaches can be distinguished; (i) grid search and (ii) random search. These simple approaches do not take any knowledge on the process that has to be optimized into account (e.g., the cost function of dimensionality reduction method). A visualisation of grid search versus random search can be seen in Figure 3-8 [109].

1-D optimization: Golden section search We assume the parameters of UMAP to be multivariate, and thus that they are dependent on each other. In this case, solving the parameter optimization one parameter at a time (e.g. 1-D optimization) is only allowed when the optimum would not vary too much for one parameter when a second parameter is being optimized. During optimization, all parameters except the examined parameter in the cycle are set to their default value. Once a parameter is optimized, that value is used in subsequent optimization cycles.

A simple framework for 1-D optimization can be constructed using the Golden section search. The derivation of the Golden section search can be found in the work of Press et al. [110],

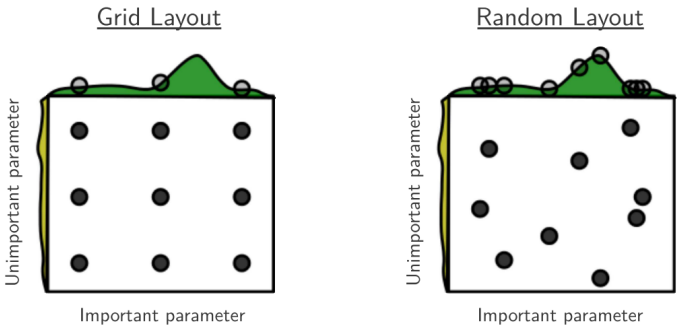


Figure 3-8: Visual comparison of grid search versus random search for a cost function determined by two hyperparameters; one is more important than the other, which is respectively shown in green and yellow on the axes. In this particular example, it can be seen that grid search is unable to detect the global optimum of the important (green) variable [109].

but the principles will be discussed here. The Golden section search does not require the function's gradient to find its optimum, only a specified interval is necessary. During every function evaluation, this range is divided using the Golden ratio (≈ 0.382). Assume we start in Figure 3-9 with bounds 1 and 2, the third point 3 is placed at fraction 0.382 between 1 and 2. This forms the triplet $(1, 3, 2)$. Next, point 4 is placed between largest fraction (between 3 and 2) using the Golden ratio (0.382) and is evaluated using function evaluation $f(x)$. Based upon the function value of 4, the triplet will change. When $f(4)$ is bigger than $f(3)$, the new triplet becomes $(1, 3, 4)$. In the opposite case, when $f(4)$ is smaller than $f(3)$, the new triplet becomes $(3, 4, 2)$. This process will repeat itself until a stopping criterion is reached. In the case of this thesis, the function evaluation is very costly since it requires evaluation of the LD embedding consisting of thousands of pixels. Press et al. use a small constant as stopping criterion, but such small value would lead to substantial runtimes of DD-UMAP. Therefore, a more aggressive stopping criterion will be discussed in section 3-4-2.

The Golden section search finds a minimum within given bounds, but finding the global optimum is not guaranteed [110]. This makes it dangerous to apply this method on functions which contain many local minimums. A possible solution can be to apply multi-start Golden section search. The parameters of UMAP can be influential to each other, possibly creating local minimums. Experiment 4 will investigate whether Golden section search can be a solution for parameter optimization of UMAP.

Grid search With grid search, the parameter space is systematically investigated by placing a grid-like structure onto said space, subsequently plugging these values into the algorithm connected to these parameters, and finally evaluating the output (as can be seen in Figure 3-8) [109]. Evaluating every parameter combination will cause an experiment to consist of $S = \prod_{k=1}^{K} \left|L^{(k)}\right|$ operations, with L the number of parameters and K the number of parameter evaluations [109]. For example, if the user would like to determine an ideal parameter configuration of an algorithm with 6 different parameters, trying 10 possibilities per parameter where each evaluation would take a second, the experiment would consist of 10^6 operations and taking over 11 days on a single workstation. This can be sped up by performing the grid search in parallel using multiple workstations, carefully registering each evaluated parameter

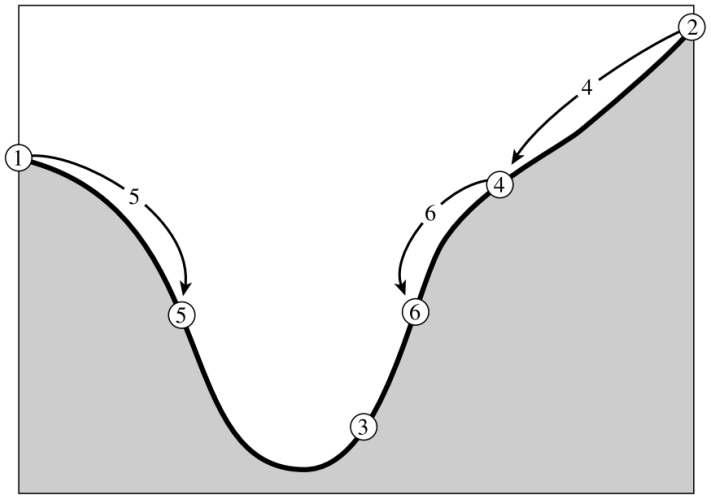


Figure 3-9: Visualization of the first iterations of the Golden section search algorithm. The initial bounds are marked by 1 and 2, and eventually this method will converge to a minimum.

configuration. The number of operations grows exponentially with the number of evaluated parameters, which makes grid search suffer from the curse of dimensionality [111]. Therefore, grid search would be only feasible on processes that are relatively fast to evaluate, and do not have too many parameters involved.

Variational autoencoders (VAE) are a deep neural network approach capable of producing a LD embedding from high-dimensional data by learning the underlying structure and compression [112]. Way and Greene introduced a VAE called Tybalt that was trained on TCGA pan-cancer RNA-seq data [112]. The parameters of Tybalt have been evaluated on synthetic scRNA-seq data created by Splatter [113] using grid search, cycling through 4 parameters of maximal 4 possibilities per parameter [114]. It was concluded that the optimized Tybalt model even outperformed t-SNE and UMAP (although the latter two methods were not optimized, and configured with default parameters). It was pointed out that the sensitivity of using different parameter settings is not widely reported in the literature [114]. The authors also mention that an unbiased approach for comparing results and evaluation of the model is important.

Random search As opposed to systematically checking every possible parameter combination, random search places the evaluated points in the parameter space completely random (see Figure 3-8). Bergstra and Bengio proved that when given the same computational budget, random search finds better models in a large configuration space of the parameters as opposed to grid search [109]. A practical benefit of random search is that it is parallelizable without needing knowledge of other workstations, since it is not necessary to keep track of which parameter configuration has been tested already [109]. Concluding; both grid and random search can be run in parallel, but grid search requires documentation of used parameter configurations while random search can be executed completely independently.

3-4-2 Sequential Model Based Optimisation (SMBO)

Using a naive optimization method can lead to unnecessary calculations in a domain of the parameter space of which it is known that no further improvement will be possible. Ideally, previous evaluations of the parameter space should be taken into account with subsequently parameter configuration evaluations. Sequential Model Based Optimisation (SMBO), also known as Bayesian optimization, takes previously taken evaluated parameter configurations into account and uses this knowledge in choosing the configuration of parameters for the next iteration. SMBO does this by optimizing an acquisition function α, trading off between exploration (e.g. sampling from areas of high uncertainty) and exploitation (e.g. sampling from areas which are likely to improve the existing model) of the parameter space, and trying to maximize these two terms [115]. A model $\hat{f}$ is constructed and extended with every newly obtained function evaluation, and aims to approximate the underlying process containing the computation of UMAP and the evaluation of the LD embedding. The model $\hat{f}$ is also referred to as the surrogate function.

These processes must be investigated carefully since too much exploration might cause the optimization algorithm to get stuck in local optimum [116]. Mathematically, a Bayesian optimization framework aims to find an input $\mathbf{x}^* \in \mathcal{X}$ to the cost function $f\colon \mathcal{X} \to \mathbb{R}$ such that $\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$. Here, we define the parameter search space $\mathcal{X}$ and the optimum input parameter configuration $\mathbf{x}^*$. Pseudo-code for a generic SMBO algorithm can be found in Algorithm 1.

Algorithm 1 Basic pseudo-code for Bayesian optimization [117]

Place a Gaussian Processes (GP) prior on f
Observe f at n_0 points according to an initial space-filling experimental design. Set $n = n_0$.
while $n \leq N$ **do** ▷ N: Total number of iterations
  Update the posterior probability distribution on f using all available data ▷ Updating model $\hat{f}$
  Let x_n be a minimizer of the acquisition function α over $\mathbf{x}$, where the acquisition function is computed using the current posterior distribution
  Observe $y_n = f(\mathbf{x}_n)$
  Increment n
end while
return Either the point evaluated with the smallest function evaluation $f(\mathbf{x})$, or the point with the smallest posterior mean.

Surrogate function Bergstra et al. applied two different SMBO versions to a hyperparameter search (e.g. surrogate functions) on deep belief networks: (i) Gaussian Processes (GP) and (i) Tree-structured Parzen Estimator (TPE) [118]. GP assumes a multivariate normal distribution of a finite number of random variables, and in order to compute the GP exactly, it unfortunately has computational complexity of $O(S^3)$ with S being the number of parameter evaluations [115]. This follows from the inversion of the covariance matrix in the computation of the mean and variance functions of the GP. The GP is made non-parametric by applying a positive definite kernel k on the input space $\mathcal{X}$ [119]. The default kernel used in the Scikit-optimize library is the Matern kernel. Bardenet and Kégl state that with a predefined GP with zero mean, any variable combination with function values $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^\top$ has a multivariate normal distribution (e.g. $\mathbf{f} \sim \mathcal{N}(0, K)$) with the covariance matrix defined as $K_{ij} := k(\mathbf{x}_i, \mathbf{x}_j)$ [119]. Then, the following mean and variance function of he GP can be obtained.

$$\begin{aligned} \mu_n(\mathbf{x}) &= k(\mathbf{x})^\top \mathbf{K}\mathbf{f} \\ \sigma_n^2(\mathbf{x}) &= k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^\top \mathbf{K}^{-1}\mathbf{k}(\mathbf{x}) \end{aligned} \tag{3-20}$$

In this equation, we can distinguish $\mathbf{k}(\mathbf{x})$ as the vector containing the covariance terms between $\mathbf{x}$ and $\mathbf{x}_{1:n}$ [115]. A GP and other Bayesian frameworks on several machine learning algorithms (latent Dirichlet allocation, structured SVMs and convolutional neural networks) have proven to surpass a human expert in selecting the hyperparameters on the CIFAR-10 data set [120].

Another strategy for the surrogate function is the Tree-structured Parzen Estimator (TPE), which approximates $p(\mathbf{x}|y)$ and $p(y)$ instead of the GP which models $p(y|\mathbf{x})$ directly [118]. Then, the TPE defines $p(\mathbf{x}|y)$ in the following way:

$$p(\mathbf{x} \mid y) = \begin{cases} \ell(\mathbf{x}) & \text{if } y < y^* \\ g(\mathbf{x}) & \text{if } y \geq y^* \end{cases}, \tag{3-21}$$

with $l(\mathbf{x})$ the density formed by using the different observations in the non-parametric densities such that corresponding loss $f(\mathbf{x})$ was less than y^*, and $g(\mathbf{x})$ is the density formed by the remaining observations [118]. It was found on Deep Belief Network (DBN) when exploring up to 32 hyperparameters, that SMBO approaches finds noticeable better points than random search and grid search by a human expert on toy data sets [118]. It has also been noted that it depends on the data set which hyperparameters are more influential on the global optimum [109]. Hyperopt [121] and Scikit-Optimize are Python libraries with several Bayesian optimizations techniques at their disposal. Hyperopt implements random search, TPE and adaptive TPE, whereas Scikit-Optimize is specialized in Sequential optimisation using decision trees and Bayesian optimization using GP. The non-parametric properties of GP and intuitive behaviour supports the choice of using GP as the surrogate function in for this thesis, using Scikit-Optimize.

Acquisition function The acquisition function α is used to score the surrogate function $\hat{f}$ and determine the next input parameter configuration for the next iteration. Several types of acquisition functions exist like Thompson sampling (TS), probability of improvement, expected improvement (EI), upper confidence bounds and entropy search (ES) [115]. EI is a popular choice for its intuitive and robust behaviour [118]. EI usually has the following form:

$$\mathrm{EI}_{y^*}(\mathbf{x}) := \int_{-\infty}^{\infty} \max\left(y^* - y, 0\right) p_M(y \mid \mathbf{x}) dy \tag{3-22}$$

with p_M the posterior GP knowing the observation history, and y and y^* respectively the surrogate values for previous and next parameter configuration estimate. Three iterations following a Bayesian optimization procedure can be seen in Figure 3-10. In this case the global objective function is maximized and the acquisition function dictates the next input parameter configuration.

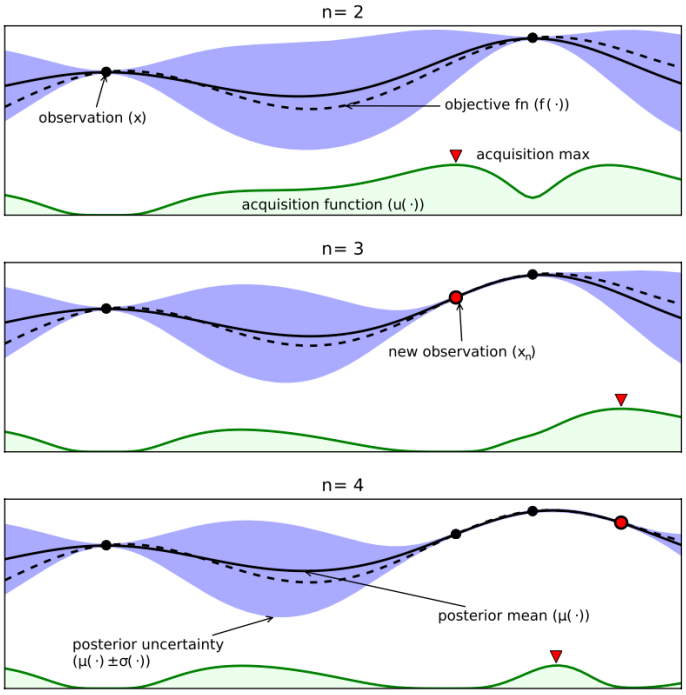


Figure 3-10: Three iterations of a Bayesian optimization framework maximizing one parameter resulting in a 1-D optimization. The observation history and probabilistic model of the objective function are shown in black lines and blue shaded regions, respectively. The true objective function is plotted as a striped curve, and will not be known in advance. It can be seen that the maximum value for the acquisition function (green) determines the next input parameter configuration [115].

Stopping criterion In order to stop the optimization algorithm in a finite amount of time, a stopping criterion should be imposed. Here, stopping criteria for both 1-D and multivariate optimization will be discussed.

In a previous study, Opt-SNE utilizes the cost function KLD of t-SNE to calibrate the early exaggeration phase, and several other parameters [12]. When no further improvement is possible, the computation is stopped. Similar stopping criterion can be implemented for the 1-D optimization framework using the Golden section search. The algorithm will switch to another parameter value when the improvement of the cost function stays for a predefined number of iterations within a predefined percentage. As the Golden section method converges to a final value for parameter p, the differences between parameter values at each optimization iteration becomes smaller and smaller. Comparing the latest values can determine when to switch from parameter values, and dictate when to switch to another parameter for further optimization. This process continues until all desired parameters are subsequently optimized.

For the SMBO approach, the number of iterations is fixed at the beginning of the algorithm. The Scikit-optimize function `gp_minimize` does not allow a variable stopping condition. In order to terminate the algorithm early, the parameter `threshold` controls the minimal number of iterations where no improvement in the minimum obtained value for the evaluation function $\text{CE}_{combined}$ is found. Whether this stopping criterion is imposed will be discussed during the construction of the experiments in chapter 4.

Chapter 4

Experiments

4-1 Data sets

During this project, three data sets have been used: (i) a synthetic data set with known spectral and spatial properties without and with additive Gaussian noise ($\sigma = 10$), (ii) a real-world data set of a mouse pup, and (iii) a real-world data set of a murine kidney. The construction of the synthetic data set will be discussed in subsection 4-1-1. The applied IMS procedure in acquiring the real-world data set will be provided in subsection 4-1-2. Finally, this section will conclude with a motivation of the chosen preprocessing steps in subsection 4-1-3, prior to performing dimensionality reduction.

4-1-1 Synthetic data set

A synthetic data set offers the benefits of knowing the exact chemical consistency of the mass spectra, and the added noise on top of these spectra. Our dimensionality reduction technique should separate pixels with dissimilar mass spectra, and cluster similar mass spectra (e.g. spectra with similar m/z intensity values). We define a spectrum P_s with a range of m/z bins from 0 to $1,000$, with maximum values for the (arbitrary) intensity peaks of 100. Each unique spectrum has 50 peaks with random locations along the spectral domain, and random intensities. Around each peak, a Gaussian distribution is placed with a variance of $\sigma^2 = \frac{16}{9} \approx 1.78$ in order to simulate a typical peak. An example of the indices with varying intensities of spectrum $P_{s,0}$ can be seen in Figure 4-1a, alongside a close-up of a peak that visualizes the isotopes in Figure 4-1b, and finally the full spectrum of $P_{s,0}$ in Figure 4-1c. Besides each pixel possessing its own spectrum, a background spectrum has been added as well, which is similarly constructed as P_s.

Two types of noise can be added when requested: Poisson and Gaussian noise. It is realistic that IMS data contains Poisson noise [40], which will cause the peaks with higher intensity to vary more than peaks with lower intensities. Furthermore, a Poisson distribution will never allow spectra to reach negative values. The resulted spectra P will take the following form:

$$P^{i,j} = P^{i,j}_{s,k} + P_b + P_n, \tag{4-1}$$

with i, j the spatial pixel location on a 2-D grid, $k \in [1, \ldots, K]$ for each unique mass spectrum with a total number of K regions with unique spectra and P_n either Poisson or Gaussian noise. The background spectrum P_b and P_n are not assigned with pixel location indices, since these will be distributed across all pixels. When applying Gaussian noise, caution should be taken to not end up with negative values within the spectrum, since this would not be possible physically. Therefore, after the Gaussian noise P_n is added on top of the generated spectra P_s, negative values will be eliminated as follows.

$$P = \begin{cases} P & , \text{ for } P \geq 0 \\ 0 & , \text{ otherwise} \end{cases} \tag{4-2}$$

Similarly to real-world data sets, the spatial coordinates of the pixels are known. Since IMS data tends to be spatial auto-correlated [40], in the synthetic data set the pixels with similar chemical consistency are thus also placed nearby each other. Four circles are defined spatially with each its own unique spectrum $P_{s,k}$ added to the final spectrum. As can be seen in Figure 4-2a, these circles will overlap with each other slightly. Within this overlapping region, we would like to simulate nonlinear mixing by multiplying a randomized fraction of the number of overlapping pixels instead of adding spectra together. In Figure 4-2b the difference between addition and multiplication of two spectra can be seen. This multiplication only affects the generated spectra P_s, Since both background spectrum P_b and the additive noise P_n is applied over all pixels as a final step. In Equation 4-3 the difference between pixels with linear and nonlinear mixing (e.g. addition and multiplication, respectively) can be seen in the form of construction of the final spectrum P for the overlapping of circles 1 and 2.

$$\begin{aligned} P^{i,j}_{add} &= P^{i,j}_{s,1} + P^{i,j}_{s,2} + P_b + P_n \\ P^{i,j}_{mul} &= P^{i,j}_{s,1} \cdot P^{i,j}_{s,2} + P_b + P_n \end{aligned} \tag{4-3}$$

Finally, each circle is unique with its own combination of either a linear gradient in the intensity of the spectrum throughout the circle, or a randomized layer affecting the whole circle. A linear gradient causes intensities to vary between $[1.0 - \alpha, 1.0 + \alpha]$ with α a variable that can be adjusted. For the random fluctuations of intensities, each pixel within a circle is multiplied by a value drawn from an uniform distribution between $[1.0 - \beta, 1.0 + \beta]$ with β again an adjustable parameter, with default settings $\alpha = 0.95$ and $\beta = 0.2$. The final spectral data P will be multiplied by the factors corresponding to the assigned circle. An overview of the different configuration per circle can be seen in Table 4-1, and also visually in Figure 4-2a.

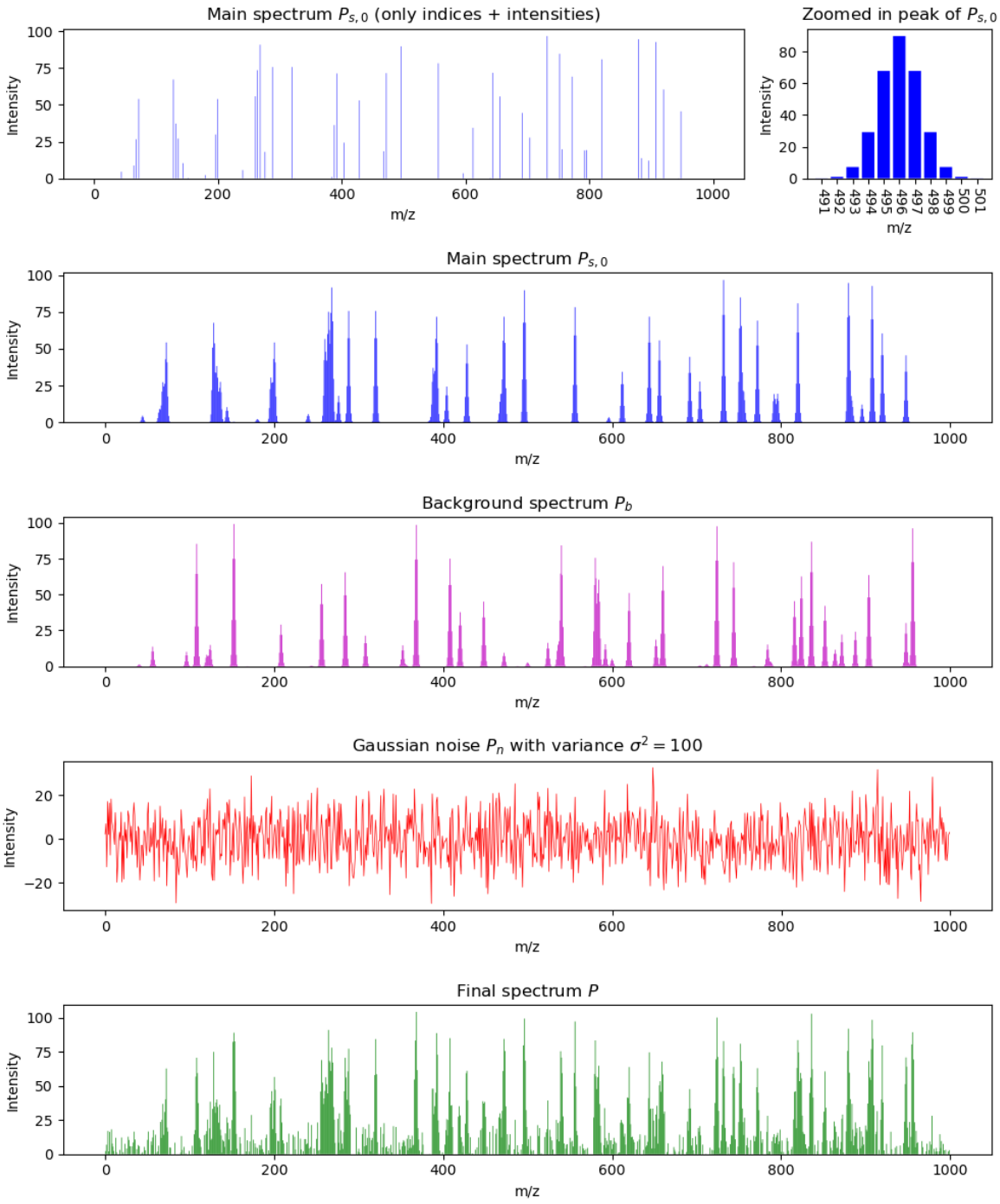


Figure 4-1: Spectral properties of synthetic data set. From top to bottom a main spectra $P_{s,0}$ is shown, with a zoomed in version of one peak to illustrate the effect of the ditributed Gaussian to simulate isotopes. A background spectrum P_b is similarly constructed to $P_{s,k}$ and shown in magenta. Generated noise (Gaussian in this case, with variance $\sigma^2 = 100$) is shown in red. And finally the final spectrum P is shown in green with only positive values as described in Equation 4-2. This shows the addition of spectra as described in Equation 4-1.

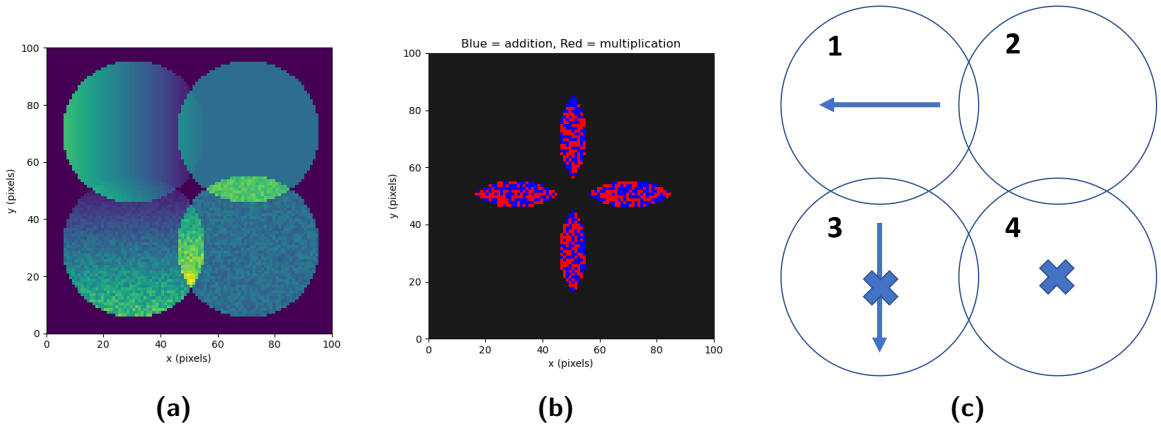


Figure 4-2: Spatial attributes of the synthetic data set. **(a)** Four circles are drawn on a grid of 100×100 pixels (total number of pixels: $N = 10,000$) with created overlap between nearby circles. Each circle contains its own spectrum $P_{s,k}$. **(b)** Here, the overlapping pixels are shown, and also whether the spectra are summed up together or multiplied by each other. The ratio of overlapping pixels between addition and multiplication is summarized in a variable `nonlinear_mixing_frac`, with is 0.5 in this case ($\rightarrow 50\%$). **(c)** The numbering of each unique circle as well as which combination of intensity variation has been applied (see also Table 4-1)

Circle	Linear gradient	Random fluctuations
1	Yes, from right to left	No
2	No	No
3	Yes, from top to bottom	Yes
4	No	Yes

Table 4-1: Properties of factors influencing the intensity per circle.

4-1-2 Real-world data set

In this thesis, two real-world data sets are investigated. Both data sets are acquired using prototype timsToF fleX mass spectrometer (Bruker Daltonik, Bremen, Germany) [122]. This subsection explains the procedure in the acquisition of the mass spectra, and motivates the taken prepossessing steps.

Mouse pup data set

The mouse pup data was acquired by Katerina Djambazova at Vanderbilt University under the supervision of Jeffrey Spraggins PhD and Richard Caprioli PhD [123]. The mouse pup images were acquired using a prototype Bruker timsToF fleX instrument (Bruker Daltonik, Bremen, Germany) [122] in trapped ion mobility spectrometry (TIMS) mode of operation with an ion transfer time of 100 μs, prepulse storage time of 8 μs, and a collision RF of 2,000 Vpp, a TIMS funnel 1 (accumulation) RF of 450 Vpp, a TIMS funnel 2 RF (analysis) of 400 Vpp, a multipole RF of 400 Vpp, and a collision cell entrance (in) voltage of 300 V. Tissue imaging data were collected at 50 μm pixel size, using 200 shots per pixel and 48% laser

power. Data were collected in positive ionization mode from m/z 300 to $1,200$. The TIMS scan time was set to 400 ms, with a reduced mobility ($1/K_0$) range of $0.4 - 1.9$ Vs/cm^2.

The mouse pup data was exported into a custom binary format optimized for storage and speed of analysis of complex IMS data. Each frame/pixel contains between $10,000 - 100,000$ centroid peaks that span the acquisition mass range and ion mobility range with $221,888$ and $4,001$ bins in the mass spectrometry and ion mobility dimension, respectively. The processing pipeline requires common dimensions along both axes, hence individual centroid peaks were inserted at their correct bin positions along both dimensions whilst missing values were set to zero. Following the conversion process, each mass spectrum and mobilogram were aligned to remove systematic instrument drift. Subsequently, a mean mass spectrum was generated, and peak picked. A total of 49 top features were selected and extracted to give the ion images that were used throughout the thesis. The full mouse data set consist of $164,808$ pixels.

Kidney data set

The murine kidney data was acquired by Elizabeth Neumann at Vanderbilt University under the supervision of Jeffrey Spraggins PhD and Richard Caprioli PhD. Rat kidney tissue was purchased from PelFreeze Biologicals (Rogers, AR, USA), sectioned at 10 μm thickness and thaw-mounted onto a conductive slide. Approximately 500 mg of DAN was sublimed at 130°C and 24 mTorr for 3.5 minutes onto the tissue surface for a final density of $\sim$ 1.0 mg/cm^2. The murine kidney images images were acquired using a prototype Bruker timsToF fleX instrument (Bruker Daltonik, Bremen, Germany) in the Q-TOF mode of operation using MALDI TIMS-IMS. Tissue imaging data were collected at 15μm spatial resolution, using 400 shots per pixel, and 35% laser power. Data were collected in positive ionization mode from m/z 200 to $1,500$.

The murine kidney data was exported into a custom binary format as described above. Data were acquired in the Q-TOF only mode, hence the ion mobility dimension is not present, in which case we introduce a secondary dimension by enforcing the data set to contain one ion mobility bin. This is carried out to ensure efficient storage and data processing without having any impact on the actual data. Following the conversion process, each mass spectrum was aligned to remove systematic instrument drift. Subsequently, a mean mass spectrum was generated, and peak picked. By filtering the mass range between m/z $500 - 900$, a total of 332 top features were selected and extracted to give ion images that were used throughout the thesis. The full kidney data set consist of $591,534$ pixels.

4-1-3 Motivation preprocessing steps

Prior to dimensionality reduction and other downstream analysis, the importance of preprocessing steps have been discussed in subsubsection 2-1-1-4. This section will discuss the steps taken on the synthetic and real-world data set with added motivation.

Since lipids are mainly of interest in the IMS data sets explored here, introducing a mass range can be useful. Lipids can be found in the region of m/z $500 - 900$ [19]. The mouse pup already is almost completely within this range, but the kidney data set contains of values up to 1600 m/z. The range of this data set will be limited to focus on lipids.

The synthetic data set has been constructed from the ground truth, using the full range of mass-to-charge ratio (m/z) bins. The assumption holds that all possible ions are measured with the chosen spectral range. We can thus apply normalization technique across the whole spectral range. The effects of difference normalization techniques have been investigated in the past [41], of which Total Ion Count (TIC) normalization is being used most frequently in the literature for its simplicity and robustness. For this reason, TIC normalization has been applied to the synthetic data set. Besides normalization, the spectrum will also be scaled using autoscaling which causes each variable to be subtracted by its mean and divided by its standard deviation [39]. It has been noted that autoscaling can suffer from the fact that noisy low-signal m/z bins can have an increased impact on downstream analysis compared to the other more prominent signals [3].

For the real-world data sets, different conditions are applicable. Normalization is not similarly possible due to the fact that the data set has been peak picked, extracting the most prominent peaks from the raw IMS data. Peak picking is a common procedure, causes missing information within mass spectra and rendering normalization often inaccurate. Therefore, the real-world data set has only been subjected to autoscaling prior to further analysis. Normalization and scaling are the last preprocessing steps, in that order.

4-2 Experiment setup

As a reminder, the thesis objective which has been formulated in chapter 1 is repeated:

> Developing an effective framework to automatically estimate the hyperparameters of Uniform Manifold Approximation and Projection (UMAP) for Imaging Mass Spectrometry (IMS) data sets with maximum usage of the available information, adaptable to different data sets, and therefore be able to extract the best possible LD embedding from the given sample in terms of its underlying (biological) structure.

In order to solve this problem, the following four questions arise. These will be investigated further in this chapter during their eponymous experiments in this chapter.

1. To what extent does the initialization phase have an effect on the produced embedding, and is there a better way of initializing than the standard procedure of Uniform Manifold Approximation and Projection (UMAP)?

2. How sensitive is UMAP to various parameter configurations? Can we classify whether a produced embedding is considered "better" than another, possibly with a combination of spectral and spatial information?

3. Is it possible to extend UMAP to incorporate spatial information during the optimization of the LD embedding, and can a better LD embedding be achieved using UMAPLUS? If so, what is the sensitivity of parameters `n_neighbors_spatial` and $\lambda_{spatial}$?

4. Is Data-Driven UMAP (DD-UMAP) able to find the optimal hyperparameters and thus produce the best possible embedding? What optimization method (1-D or Bayesian optimization framework) is more suitable to find the global optimum in the defined parameter search-space? And how does DD-UMAP compare to traditional grid search?

Unless stated otherwise, all parameters of UMAP will be configured with the default values. Before any experiments can take place, the subsampling of the pairwise distance matrices during the evaluation of the LD embedding will be tested, with as constraint the physical memory of the workstation. This will result in a numerical value $\text{CE}_{combined}$ allowing scoring the produced LD embedding. Formally, this will be called Experiment 0, and its findings will be discussed in the next chapter before all other experiments.

4-2-1 Experiment 0: Subsampling

During evaluation of the LD embedding, the pairwise distance matrices X and Y are being used. Since the size of these matrices scales quadratically with the number of data points N, this becomes problematic for large data sets ($N = 10^4$ requires 1.60 Gb of RAM, whereas $N = 10^6$ requires 16.0 Tb of RAM). Since real-world IMS data sets can consist in the order of 10^6 data points (e.g. pixels), the proposed evaluation method in Equation 3-16 would fail. Therefore, this experiment explores the possibilities of subsampling the data in order to hold the entire pairwise distance matrix in memory while preserving the reliability of using the chosen evaluation method. Two types of constraints will be differentiated: (i) physical (such as memory usage) and (ii) the total runtime of the evaluation method. The physical limitation chosen for this thesis is a total memory usage of 2.00 Gb, which leads to $N_{max} = 11,180$ pixels. This choice is based on the specifications of the workstation. Prior to this experiment, no runtime constraints are added.

During this experiment, both synthetic data and the mouse pup are being used. Since the synthetic data set contains $N = 10,000$ data points, this falls within the memory constraints. The mouse pups consist of $N = 164,808$ pixels, and thus only up to $N_{max} = 11,180$ pixels are being used. Using random subsampling with uniform distribution, a selection of the total number of pixels is taken. `n_iter_eval` $= 10$ number of evaluations using their own unique subsample of the data are being computed. The performance of the evaluation of the LD can be measured by the standard deviation σ of multiple evaluations. Subsequently, the standard deviation $\sigma_{multiple}$ can be expressed as a percentage of the average $\mu_{multiple}$, which will be denoted as the confidence score ϕ. When ϕ is below a certain percentage, we can rely on the produced value for μ ($\text{CE}_{combined}$), and it is used as scoring metric for every other experiment.

4-2-2 Experiment 1: Initialization phase

By default, UMAP initializes its LD embedding using `init` = "spectral", which indicates the construction of the kNN graph with the normalized Laplacian matrix as described in Equation 3-10. Besides the spectral initialization, the randomized initialization is available in UMAP. The algorithm falls back to randomized initialization whenever the kNN graph is not locally connected and the construction of the normalized Laplacian is not possible. Finally

the spatial initialization is proposed using the original pixel coordinates. In subsection 3-1-1, these different initializations have been described in more detail.

Kobak and Linderman argue that both t-SNE and UMAP would preserve global structure poorly when both algorithms are randomly initialized, and that beside the initialization not much separates both dimensionality reduction techniques [96]. In the study of Kobak and Linderman, only the spectral and random initialization are considered. It raises the following question:

To what extent does the initialization phase have an effect on the produced embedding, and is there a better way of initializing than the standard procedure of Uniform Manifold Approximation and Projection (UMAP)?

This experiment will try to answer this question by subjecting UMAP to 3 types of initialization: (i) spectral, (ii) random and (iii) spatial initialization. All parameters of UMAP will be configured to their default settings, except `n_components` and `n_epochs`. When varying the dimension of the LD embedding (e.g. `n_components` or K) for the spatial initialization, all dimensions of $K > 2$ will be set to zero, since all IMS data sets being used in this thesis are spatially 2-D and initializing in 1-D would not make sense. We would like to visualize the evolution of UMAPs optimization phase over time for different initializations with varying K.

4-2-3 Experiment 2: Investigation of cost function and hyperparameter configuration

In subsubsection 3-3-2-1 the hyperparameters of the UMAP algorithm are summed up. The complete list of parameters can be found in section A-1. The usual approach for most papers incorporating UMAP is to use the default values for these parameters, or change at most the distance metric. This leaves potential for further improvement of the LD embedding. It raises the following question:

How sensitive is UMAP to various parameter configurations? Can we classify whether a produced embedding is considered "better" than another, possibly with a combination of spectral and spatial information?

This experiment aims to map the parameter space of UMAP to give recommendations for which parameters to tune during experiment 4 in subsection 4-2-5, since not all parameters within UMAP yield sufficient improvement on the LD embedding. The introduction of UMAPLUS brings rise to $\lambda_{spatial}$, which has been proposed to be included into the loss function to evaluate the LD embedding in Equation 3-16. The LD embedding will be visually assessed to make sure $\text{CE}_{combined}$ is operating properly and can be relied on in an unsupervised manner. Visualization of the embedding for $K = 2, 3$ is possible with a continuous RGB colormap. For higher dimensions of the LD embedding, cluster algorithms can assign colors to the pixels. However, the assigned colors will be discrete; a continuous colormap will not be possible for $K > 3$. For visualization purposes, the dimension of the LD embedding will be kept at $K = 2$ during the semi-grid search, so we can identify the structures in the embedding on a 2-D visualization. This will also motivate the usage of spatial information, since this information can only be stored in two dimensions for most regular IMS data sets which are acquired from sampling a tissue surface.

The choice for the distance metric used for the construction of the kNN-graph $\mathcal{G}_{spectral}$ on the original high-dimensional data remains a critical objective. During the semi grid search the distance metric will be varied alongside the one parameter to observe changes with the default Euclidean distance metric within UMAP. These changes can be registered visually by investigating or comparing the LD embedding and the corresponding image domain, or by plotting the CE.

When everything is operating properly, the ultimate goal of this experiment would be to rely on the output value of some evaluation measure (such as $\text{CE}_{combined}$) for the evaluation of the LD embedding produced by UMAP. We will investigate whether the addition of using spatial information in the cost function is valuable, or if we should rely more on spectral information only. A semi grid search will explore parameter space of UMAP on the synthetic data set with additive Gaussian noise ($\sigma = 10$).

4-2-4 Experiment 3: Comparing UMAP with UMAPLUS, using spatial information

In subsection 3-3-1 UMAPLUS has been introduced, a modified version of UMAP forcing neighboring pixels to "stick" together based on a new adjustable parameter $\lambda_{spatial}$ and the number of neighbors `n_neighbors_spatial` for the construction of $\mathcal{G}_{spatial}$. This experiment will try to answer the following question rising along with the creation of $\lambda_{spatial}$:

Is it possible to extend UMAP to incorporate spatial information during the optimization of the LD embedding, and can a better lower dimensional embedding be achieved using UMAPLUS? If so, what is the sensitivity of parameters `n_neighbors_spatial` *and* $\lambda_{spatial}$*?*

Besides varying $\lambda_{spatial}$ and `n_neighbors_spatial`, also the number of neighbors for the construction of the kNN graph $\mathcal{G}_{\text{spectral}}$ will be varied. In UMAP, this parameter is called `n_neighbor`, but will be referred to as `n_neighbor_spectral` during this experiment in order to remove any confusion with `n_neighbor_spatial`.

4-2-5 Experiment 4: Automated searching of parameter space (DD-UMAP)

Finally, the results of previous experiments will be combined to formulate the optimization method for DD-UMAP. During this thesis, two optimization methods have been described in section 3-4: (i) 1-D optimization using Golden Section Search and (ii) a multivariate approach described by a Bayesian optimization method. Each method came with its advantages and disadvantages, but the following questions remain the central objective during this experiment.

Is Data-Driven UMAP (DD-UMAP) able to find the optimal hyperparameters and thus produce the best possible embedding? What optimization method (1-D or Bayesian optimization approach) is more suitable to find the global optimum in the defined parameter search-space? And how does DD-UMAP compare to traditional grid search?

First, a list of most prominent parameters have been discussed in subsubsection 3-3-2-2 and is further investigated in subsection 4-2-3. The final list of parameters which will be optimized during this experiment and their value range is empirically determined and distilled in Table 4-2.

Parameter	Value range $[min - max]$	Data type
`spatial_lambda`	$[0.01 - 1.0]$	Float
`n_neighbors`	$[2 - 100]$	Integer
`n_neighbors_spatial`	$[4, 12, 28, 48, 80, 112, 148, 196]$	Categorical
`learning_rate`	$[0.1 - 10.0]$	Float
`min_dist`	$[0.0 - 1.0]$	Float
`spread`	$[1.0 - 3.0]$	Float
`set_op_mix_ratio`	$[0.1 - 1.0]$	Float

Table 4-2: List of UMAP's parameters which will be optimized during this experiment. The second column lists the allowable value range of the parameter. However, for parameters of the type *Categorical*, this indicates the multiple allowable binary states of said parameter.

1-D optimization The two possible optimization methods will both be applied in DD-UMAP. The 1-D optimization method is finding its next optimal point using Golden section search, as described in subsection 3-4-1. Using this technique, the parameters of UMAP will be investigated one by one, starting with the most prominent ones. All other parameters will be configured to their default value. From here, we can make a few choices in the optimization algorithm. We can use the found optimum value of the parameters, which has been optimized during further evaluation steps, or we can start every search with all parameters at their default value. Starting from their default values would cause us the rely on the default choice of the authors of UMAP for said parameter [11], and a potential optimum could be missed.

After every evaluation, UMAP can be initialized by 3 choices discussed in subsection 3-1-1, or the produced embedding from the previous iteration can be plugged into UMAP again for further optimization of the LD embedding with a different parameter configuration. The latter can help less influential parameters fine tune the embedding better than starting from scratch. However, since we aim to search through the parameter space in an organised matter and remove as much dependencies as possible, the spectral initialization will be used at every optimization cycle.

In order to stop the Golden section search in a finite amount of time, a stopping criterion will be introduced as already discussed in section 3-4-2. Golden section search requires either a set number of function evaluations `n_iter` or a threshold of deviation of the parameter value θ_{param} or the CE value θ_{CE} (e.g. *percentage of improvement*) in order to terminate its search. For robustness, we expect these values to change minimally, and introduce a minimum number of iterations Δ_θ to be under these thresholds for the algorithm to finally terminate and move on to the following parameter configuration. We define values for these thresholds in Table 4-3.

The procedure of the Golden section search is explained in section 3-4-1. This method is unable to deal with categorical data such as `n_neighbors_spatial`. Therefore, during the 1-D optimization, this parameter will be considered as an integer with bounds between $[4, 196]$. This results in uneven circles of neighboring pixels. An even number of neighboring pixels has been illustrated Figure 3-4.

Random search Next to a grid search, we introduced a random search through the selected parameter space in section 3-4-1. Previous studies have proven to give similar or better results

Parameter	Value
`n_iter`	20
θ_{param}	3%
θ_{CE}	5%
Δ_{theta}	3 iterations

Table 4-3: Settings for the 1-D optimization framework.

for the hyperparameter optimization with random search opposed to a grid search [109]. The infeasibility of a grid search has been discussed, yet a random search provides the possibility of exploring the entire parameter space. Practically, this implies that we initialize UMAP with random parameter configurations within the range described in Table 4-2.

Bayesian optimization Lastly, the Bayesian optimization framework will be investigated. This framework is utilizing Gaussian Processes (GP) for the construction of the surrogate model $\hat{f}(\mathbf{x})$. Further choices for the construction of this framework have been discussed in subsection 3-4-2. The Python library Scikit-Optimize will be used for construction of this method, using a Matern kernel and the expected improvement (EI) as acquisition function.

The parameters as input of the optimization loop will be divided in 3 groups: (i) small, (ii) medium and (iii) large. The small group will only optimize 1 parameter and can be compared best with the naive 1-D optimization framework. The expectation is that the Bayesian optimization approach performs equal or better than the 1-D optimization method. The medium group will consist of 3 parameters: `n_neighbors`, `n_neighbors_spatial` and $\lambda_{spatial}$. This parameter space is explored using grid search in experiment 2. Finally, all parameters mentioned in Table 4-2 will be the input for the large framework.

The optimization will be initialized by a number of random function evaluation `n_initial_points`, followed by a number of more intelligent iterations `n_intelligent_points`. The choice for these settings is based on the number of parameters which we want to optimize, and will thus vary based on the multiple groups with a different number of parameter in its optimization framework.

Using the Scikit-optimize library, the optimization will always take a fixed amount of iterations `n_calls`. This can be modified such that the algorithm will be terminated earlier when a stopping criterion is met. The minimal value found during function evaluation is saved after each iteration. When this value does not change for a variable number of iterations, $\Delta_{B.O.}$, and `n_calls` is not reached, the optimization can be terminated. This eliminates wasting of valuable computing power on iterations in parts of the parameter space that are unfruitful. However, during this experiment, we are curious to see how long it will take to achieve a better parameter configuration for long runs. Therefore, this stopping criterion will be turned off during this experiment in order to observe the long term converge around the optimal parameter configuration.

4-2-6 Case study: Assessment of real-world IMS data sets

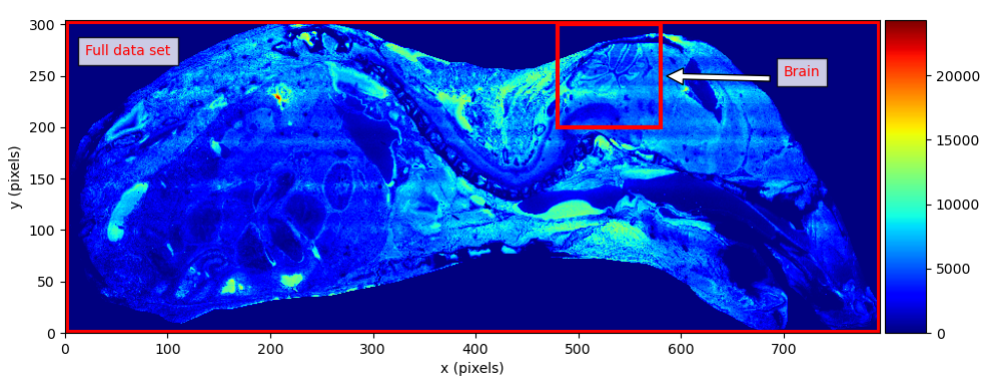
In a separate case study, both UMAPLUS and DD-UMAP are applied to two real-world IMS data sets: (i) a mouse pup and (ii) a murine kidney. These data sets contain substantially

more pixels than the synthetic data set, as has been discussed in subsection 4-1-2. First, subsections from the data sets will be considered. The brain in the mouse pup data set showed already some interesting structure when inspecting the ion images, so this will be the first considered subsection. The kidney data set contains less structure, which makes it more difficult to place the subsection to visualize interesting elements. A study by Tideman et al. identified three ROIs in the kidney data set; renal inner medulla, outer medulla, and cortex [124]. The first subsection will be placed on the cortex. Since the structure of the kidney data set is less present, we expect UMAPLUS to be more effective than regular UMAP.

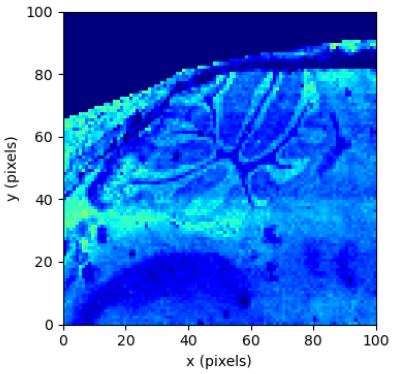
Secondly, a larger part of the data set is considered. The whole mouse pup is investigated, and the right half of the kidney data set containing all substructure (renal inner medulla, outer medulla, and cortex). The number of pixel of the subsection is $N \approx 10,000$ and for the larger sections $N \approx 160,000$. Computing the confidence score ϕ will motivate usage of a subset of the data for evaluation.

Both the 1-D optimization method and the Bayesian approach are applied to the data sets. The large framework is considered for optimizing 7 parameters of UMAP (see Table 4-2). While we used the spectral initialization during Experiment 4, now the optimization methods are also started with the spatial initialization. All combinations provide for a total of 4 evaluation scores and thus 4 parameter configurations per evaluated part of data.

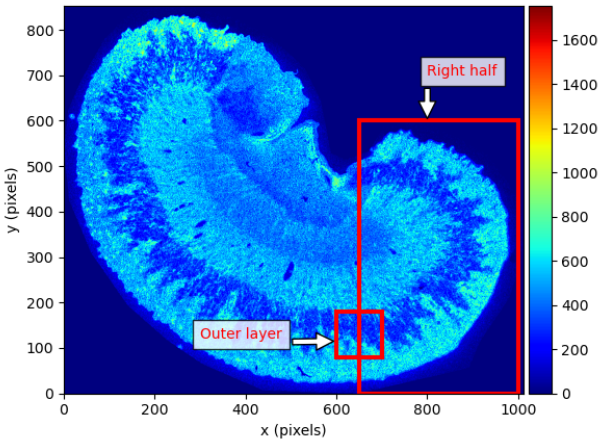
Finally, the best version of either spectral and spatial initialization using both optimization techniques are compared to several existing dimensionality reduction techniques: PCA, t-SNE and UMAP with default settings. The resulting LD embedding and image domain are visualized with a target dimension of 2 ($K = 2$), and using the RGB-colormap.



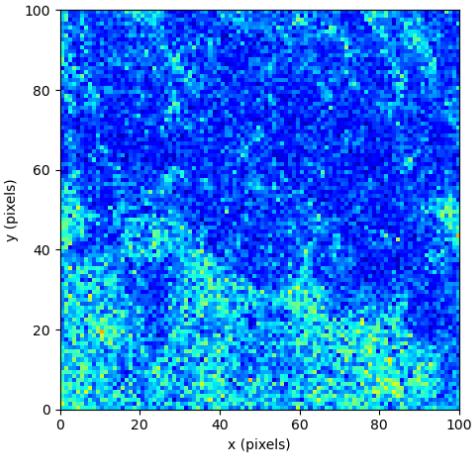
(a) Full mouse pup data set



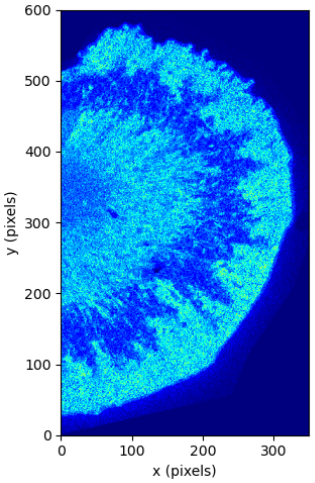
(b) Subsection: Brain



(c) Full murine kidney data set



(d) Subsection: Outer layer



(e) Subsection: Right half

Figure 4-3: Visualization of the ion images corresponding to the two real-world data sets and the selected subsection for further investigation. The ion images belonging to the mouse pup and murine kidney are from m/z 783.590 and m/z 524.373, respectively.

Chapter 5

Results and Discussion

This chapter will evaluate the described experiments in previous chapter.

5-1 Experiment 0: Subsampling

Before any experiment will be worked out, we make sure the evaluation function formulated in Equation 3-16 is allowed to be used when subsampling the data. The standard deviation σ can be expressed as a percentage of the average value μ over multiple evaluations (`n_iter_eval`) using the same fraction, with each evaluation having its own unique subset of the data. This shows merely the deviation of the produced evaluation function $\text{CE}_{combined}$. In order to rely on μ, multiple runs are executed (`n_iter_eval_multiple` = 10), producing multiple μ giving rise to $\mu_{multiple}$ and $\sigma_{multiple}$ (the average and standard deviation of multiple μ). These values are combined to form a confidence score ϕ.

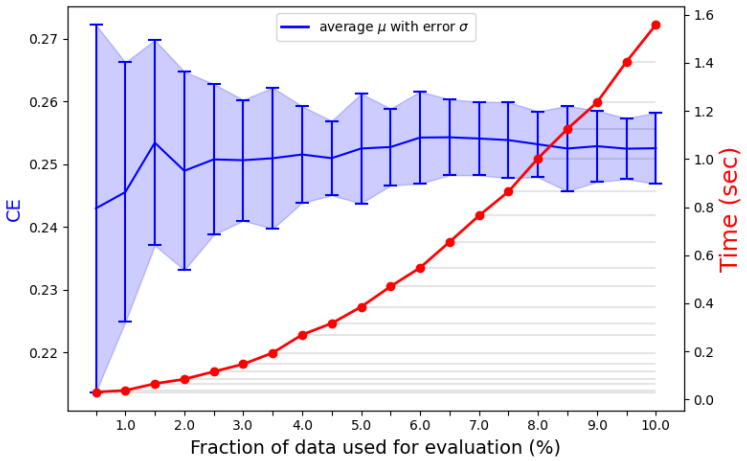
Synthetic data set First, the synthetic data is considered with $N = 10,000$. Different fraction of the data with `n_iter_eval` = 10 are evaluated, and their values for μ, σ and the time taken for the evaluation of that fraction are plotted in Figure 5-1. Whether the produced value is correct is not the topic of this discussion. As more data is being used for evaluation, the variance decreases, and is zero when all data is used. It can clearly be seen that the evaluation runtime per fraction of the data grows exponentially with the increase of fraction of the data.

Similarly to the runtime, the allocated memory size scales quadratically as can be seen in Figure 5-1c and Figure 5-1d. Here the confidence score is defined as $\phi = \frac{\sigma_{multiple}}{\mu_{multiple}} 100\%$, with 10 times μ using different `random_state` as input data. The confidence score gives in this context the reassurance of using a certain subset of the data and still relies on the produced μ by $\text{CE}_{combined}$. The confidence score is not given as feedback during regular evaluation. Going back to Figure 5-1c and Figure 5-1d, a CE percentage bound of 1% is considered *good enough*. This motivates a choice to select only 10% of the synthetic data set with additive

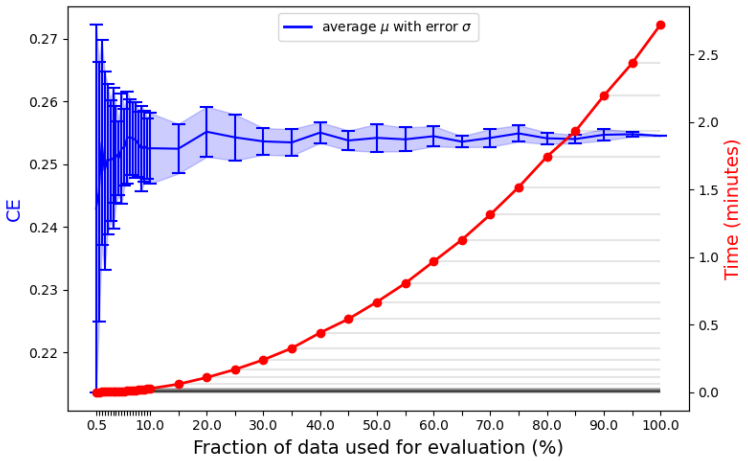
Gaussian noise for evaluation to still be adequately confident in the average $\text{CE}_{combined}$ value μ.

Two constraints can be predefined: (i) physical and (ii) runtime constraints. The physical constraint is usually satisfied sooner, and is imposed as a hard constraint. The used memory may not exceed 2.0 GB. The entire run using the synthetic data set fits within this limit, but it becomes problematic for real-world data sets consisting of over $100,000$ pixels. The additional benefit of the memory constraint allows the evaluation phase to speed up significantly; from 2.8 minutes using the entire data set to 1.6 seconds using 10% of the data. For now, the runtime constraint is imposed as a soft constraint, e.g. shorter runtimes are more feasible. In the future, weights may be added to the runtime in order to penalize longer runtimes.

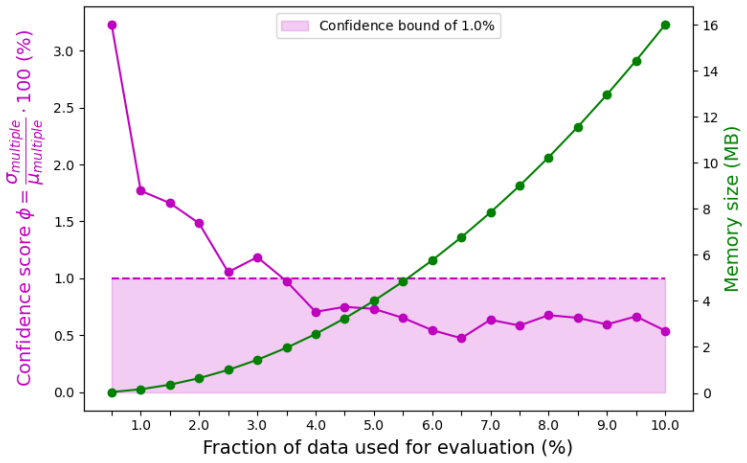
Real-world data set; mouse pup Secondly, the entire mouse pup data set is evaluated on subsampling for evaluation. This time, the physical constraint has been implemented since $N = 164,808 > N_{max} = 11,180$ and subsequently not all data can be used at once for evaluation. The average values μ and standard deviation σ for different fractions of the data have been plotted in Figure 5-2. It is impossible to evaluate 100% of the data using the proposed evaluation method due to the required memory usage, so only up to 5% will be evaluated. Following a similar procedure with the confidence score as on the synthetic data set, not less than 1.25% of the data should be used for evaluation to be confident enough in the produced μ as $\text{CE}_{combined}$. It can be concluded that subsampling makes it possible to evaluate large IMS data sets without losing too much information through predefined bounds.



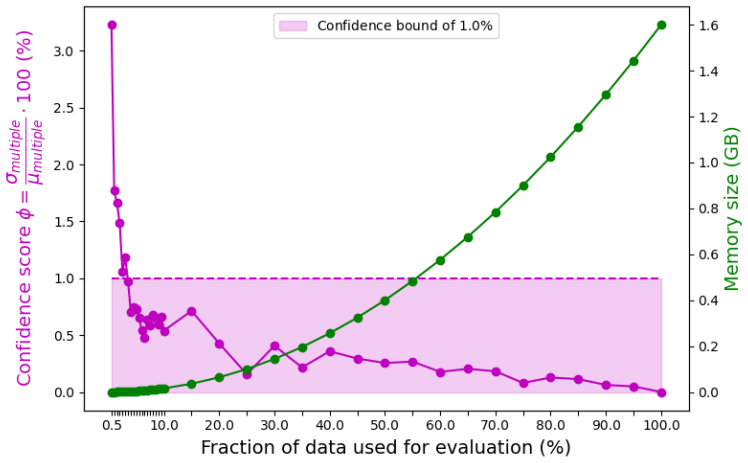
(a) Zoomed in



(b) Full range

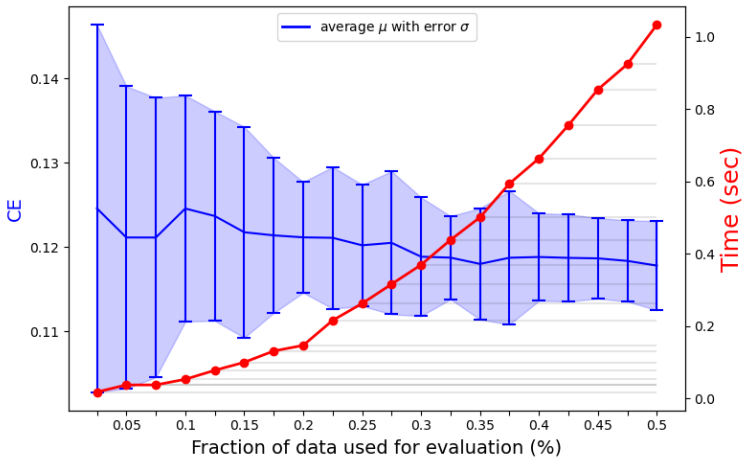


(c) Zoomed in

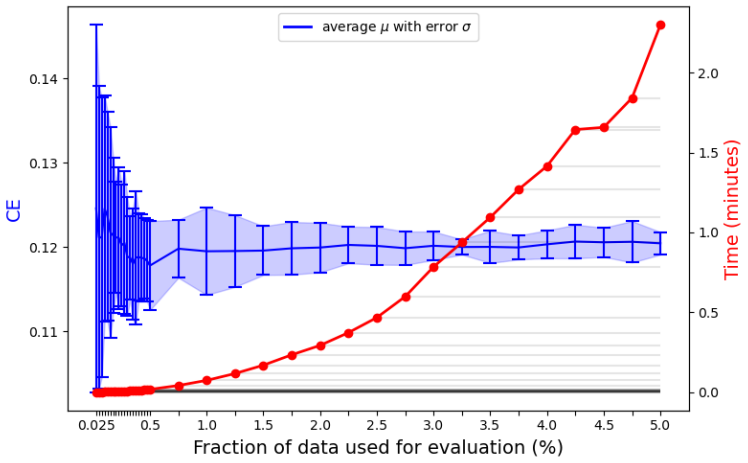


(d) Full range

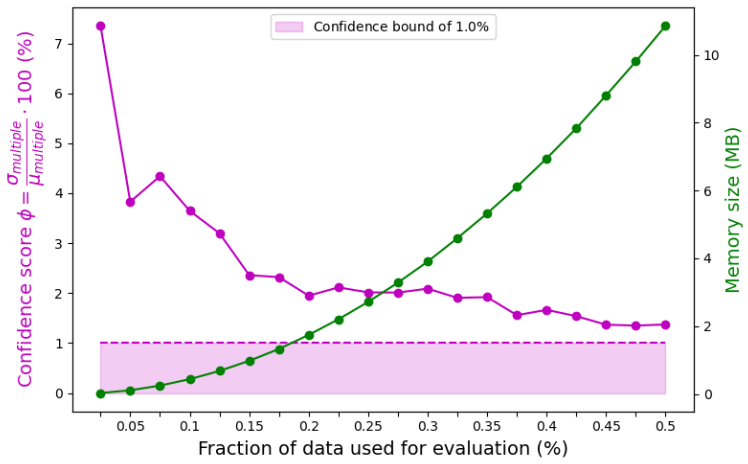
Figure 5-1: Subsampling of the synthetic data with additive Gaussian noise ($\sigma = 10$) using different fractions for evaluation of $\text{CE}_{combined}$. **(a) & (b)** Each fraction consist of 10 runs, giving the average μ as final value for $\text{CE}_{combined}$ with a variance expressed as the standard deviation σ. The variance converges to zero when increasing the fraction of the data used for evaluation. The runtime of all 10 runs combined is expressed in red, growing exponentially. **(c) & (d)** The confidence score when running multiple evaluations (with each evaluation containing 10 runs) in order to motivate using a certain fraction of the data used for evaluation. A variance bound of 1% is imposed to validate the usage of μ as $\text{CE}_{combined}$. The size required in memory is plotted in green, scaling exponentially.



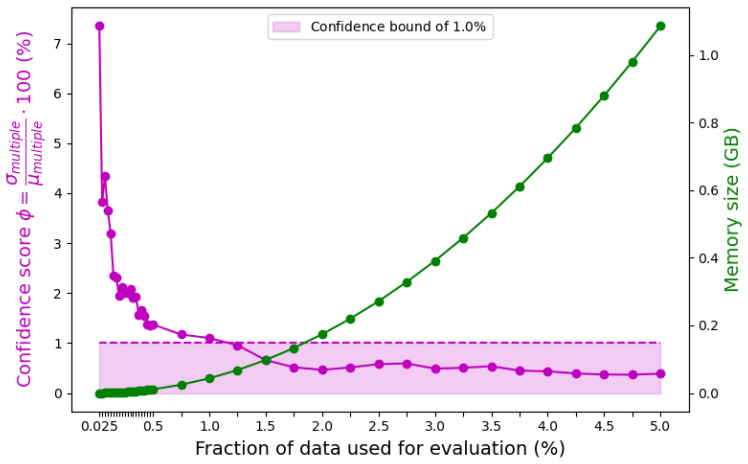
(a) Zoomed in



(b) Full range



(c) Zoomed in



(d) Full range

Figure 5-2: Subsampling of the entire mouse pup data using different fractions for evaluation of $CE_{combined}$. **(a) & (b)** Each fraction consist of 10 runs, giving the average μ as final value for $CE_{combined}$ with a variance expressed as the standard deviation σ. The variance converges to zero when increasing the fraction of the data used for evaluation. The runtime of all 10 runs combined is expressed in red, growing exponentially. **(c) & (d)** The confidence score when running multiple evaluations (with each evaluation containing 10 runs) in order to motivate using a certain fraction of the data used for evaluation. A variance bound of 1% is imposed to validate the usage of μ as $CE_{combined}$. The size required in memory is plotted in green, scaling exponentially.

5-2 Experiment 1: Initialization phase

With the justification for subsampling of the data in the previous experiment, we continue to investigate the initialization phase. As explained during the last chapter, the spectral, random and spatial initialization will be assessed.

Varying `n_components` and `n_epochs`

We define the number of computations within UMAP as the total amount of force applied to the kNN-graph $\mathcal{G}_{spectral}$ during the optimization of the LD embedding. It has to be noted that the number of computations within UMAP does only scale with the parameters `n_neighbors` (since more vertices in graph $\mathcal{G}_{spectral}$ result in computation of more forces) and `n_epochs`. When varying the dimension of the LD embedding (`n_components` or K), as will be executed in this experiment, the number of computations remain constant. However, the degree of freedom grows with increasing K. This allows relaxation on the underlying manifold in the data.

Here, we compare the spectral and spatial component of Equation 3-16, as well as the combined value ($\text{CE}_{combined}$). First, the synthetic data set will be evaluated with default settings and additive Gaussian noise ($\sigma^2 = 10$). In Figure 5-3a both spatial and spectral components are visualized for a spectral initialization, as well as the combined values following Equation 3-16.

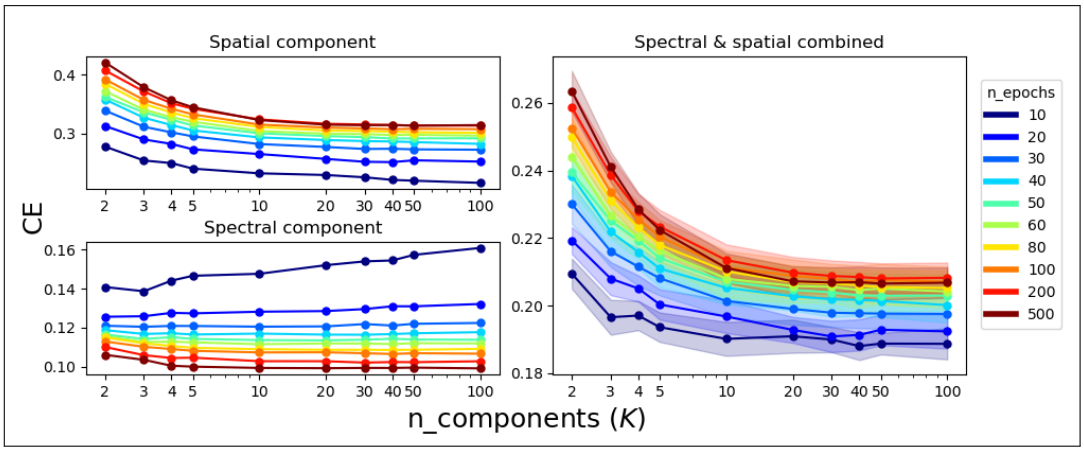
No further improvement seems to occur for dimensions higher than $K = 5$, although the difference is minimal. Due to the increase of the degrees of freedom but the stagnation of the number of computations within UMAP, not much is gained by searching in higher dimensions.

This effect becomes strongly visible in Figure 5-3b, which visualized the random initialization. For a low number of epochs, the necessary amount of computations for the optimization of the LD embedding is not finalized for $K > 5$. The degree of freedom is simply too large for the allowable number of computations. Interestingly, the spatial component shows improvement at this point. It must be noted that the spatial information is not used by UMAP (e.g. $\lambda_{spatial} = 0$). Comparing the spectral with the spatial initialization shows the more stable performance of the LD embedding for $K > 5$ using the superior normalized Laplacian matrix for the initial pixel locations compared to a naive random initialization.

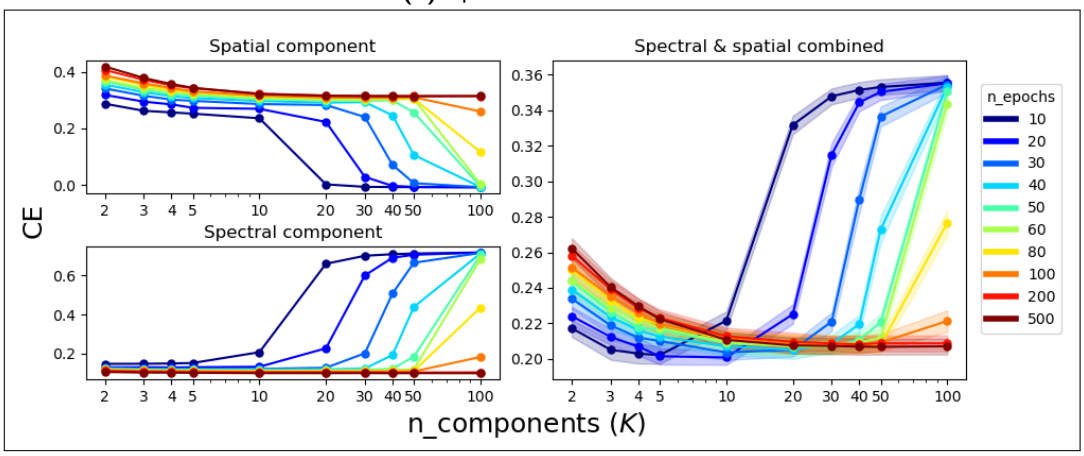
Interestingly, the spatial component with the random initialization improves for a lower number of epochs. This comes from the fact that during a longer optimization, the formed clusters in the LD embedding will become tighter. This is less feasible when the scoring metric is based upon the pixel locations, which favors the data points in the embedding to be spread out. This can be less favorable for extracting well defined structures from the embedding since clusters of dissimilar chemical consistency will be closer to each other.

The performance of the spatial initialization can be seen in Figure 5-3c. Since all dimensions above $K = 2$ are set to zero, increasing `n_components` is practically useless as no forces can be applied in the direction of the null space. Subsequently, the performance over all K of the LD embedding is very similar.

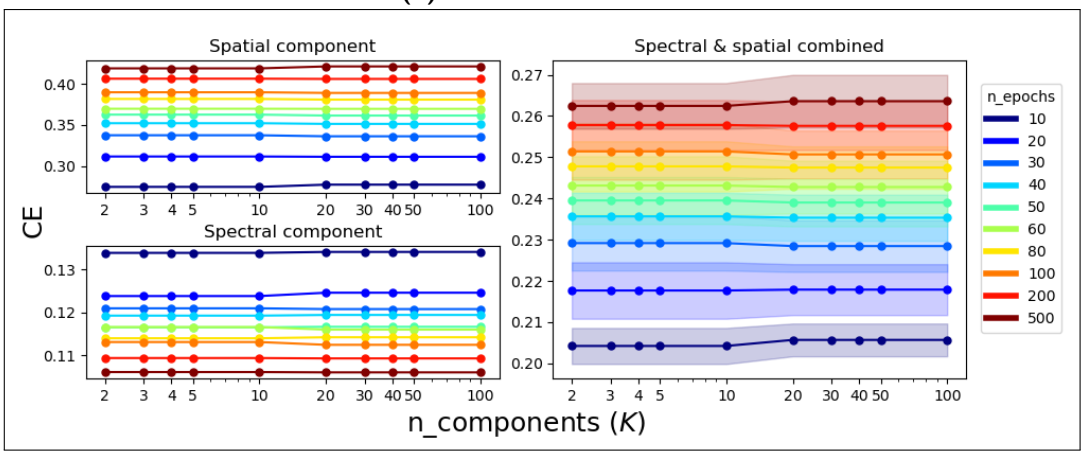
While choosing the correct K for the LD embedding is important for determining the degrees of freedom of the underlying manifold, for visualization purposes the embedding is mostly



(a) Spectral initialization



(b) Random initialization



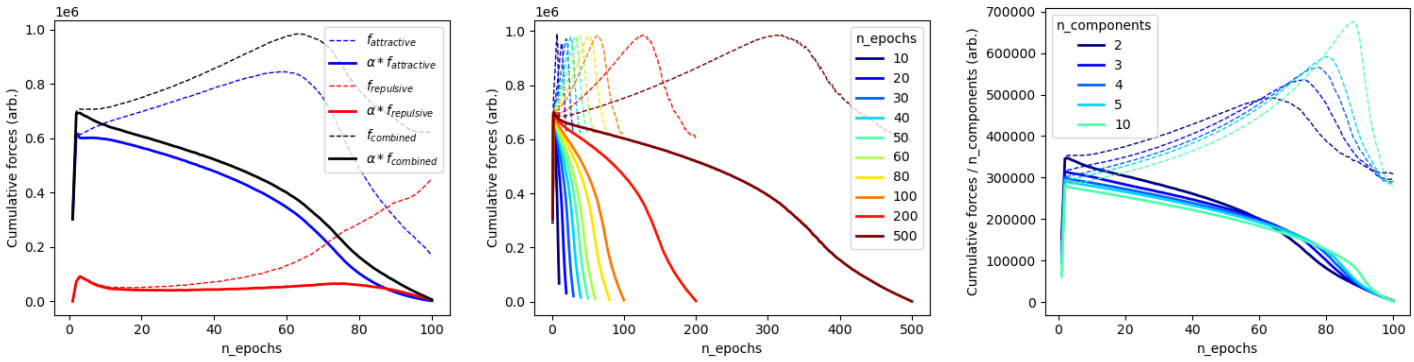
(c) Spatial initialization

Figure 5-3: Visualization of spectral, random and spatial initialization for varying `n_components` and `n_epochs`. Both separate values for the spectral and spatial component of Equation 3-16 are visualized, as well as the full $CE_{combined}$. The filled in region resembles the standard deviation of $CE_{combined}$.

limited to 2 or 3 dimensions. In these dimensions, the added benefit is the visualization using a continuous RGB colormap with $K = 3$ and a RG (red-green) colormap with $K = 2$.

Forces within UMAP

UMAP optimizes its LD embedding by placing forces on the edges of the constructed kNN-graph $\mathcal{G}_{spectral}$, as explained in section 3-1. A distinction is made between attractive and repulsive forces, $f_{attractive}$ and $f_{repulsive}$ respectively. All forces computed by UMAP at each individual epoch with `n_epochs` $= 100$ and `metric` $=$ *cosine* have been summed up, and their cumulative value can be seen in Figure 5-4. Clearly, the biggest impact on the kNN comes from $f_{attractive}$, which follows from the default setting of the ratio of attractive/repulsive sampling rate (`negative_sample_rate`). It takes UMAP a few epochs to get to full power, after which the learning rate α forcefully pushes all forces to zero, thus creating a stable LD embedding. As can be seen in Figure 5-4b, the amount of forces scales linearly with `n_epochs`. Figure 5-4c shows that the normalized forces does not increase with `n_components`. In this case, the forces scale with the number of dimensions of the LD embedding.



(a) Single cycle of UMAP **(b)** Varying `n_epochs` **(c)** Varying `n_components`

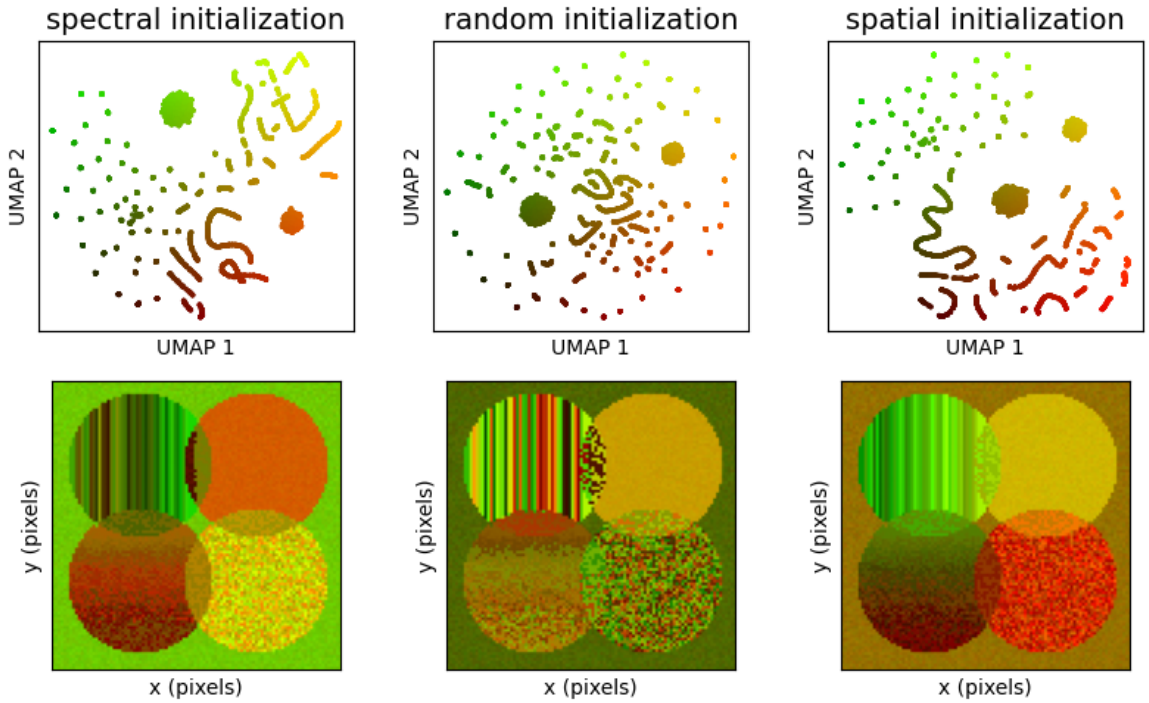
Figure 5-4: Visualization of the forces applied by UMAP, cumulatively added per iteration. Data set: synthetic with additive Gaussian noise ($\sigma = 10$). **(a)** The truly computed forces are plotted in striped lines, but the learning rate α forces them to zero in a predefined number of epochs; `n_epochs`. **(b)** Varying `n_epochs` does not show an increase in the total amount of forces applied to the kNN-graph. **(c)** This subplot shows the normalized cumulative forces per `n_components` for varying `n_components`.

Comparing LD embedding and image domain

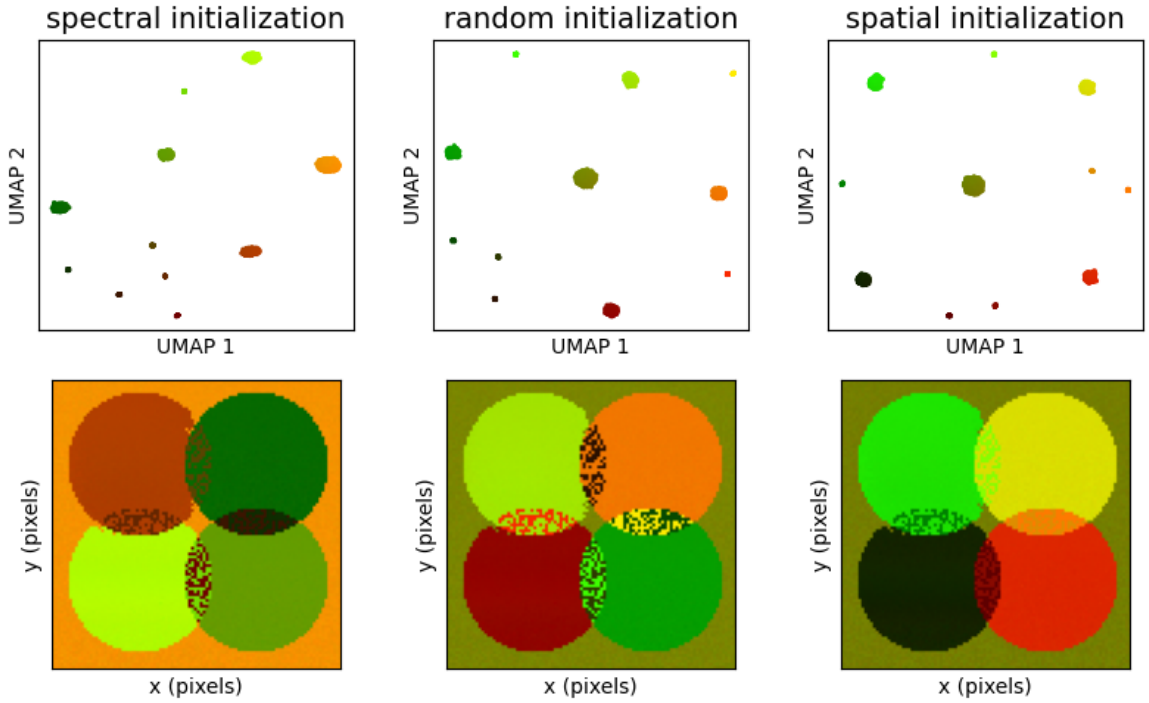
First, the synthetic data with additive Gaussian noise ($\sigma = 10$) is considered. During spectral initialization in Figure 5-5b, the nonlinear mixing is clearly visible and shows the multiplied spectra positioned close to each other in the LD embedding, indicating that the spectral initialization renders these pixels similar. However, we know that the overlapping multiplied pixels are not similar to each other, but are indeed very different from the additive overlapping pixels. During random initialization, the nonlinear mixing is not handled any better. A more intuitive visualization can be seen with the spatial initialization since the LD embedding and image domain are conveniently aligned. In the LD embedding, the 4 circles of the synthetic

data set are positioned identically to the positions in the image domain. Even the overlapping pixels are close together and where we expect them in the LD embedding, and even in these clusters we can identify the additive and multiplied pixels clearly. This shows the strength of the spatial initialization; for $K = 2$, the LD is intuitive and creates a clear explanation of the data.

Secondly, the noiseless synthetic data set is considered, which is visualized for different initializations in Figure 5-5a. Since this time the gradient in the left circles is not overruled by the additive noise, a clear separation of the vertical lines can be seen. The spectral initialization is preserving the gradient in the upper left circle adequately, but the bottom left circle is having trouble in exposing its gradient due to the additive noise in this circle. The random initialization performs far below average with a gradient of the upper left circle being unrecognizable and the other circles being very noisy. Again, the spatial initialization shows superiority with a clear gradient in both circles. The image domain is the most clean of the three initializations, and the LD embedding is once again recognizable. The connection with the noisy data set is that both image domain plots using spatial initialization are positioned at similar positions in the LD embedding, and thus consist of a similar colormap. This makes comparison of the different embeddings more practical than any other initialization, and works since the data exhibits a clear spatial structure.



(a) Synthetic data set with additive Gaussian noise ($\sigma^2 = 10$).



(b) Synthetic data set without noise.

Figure 5-5: Comparison of three different initializations: (i) spectral, (ii) random and (iii) spatial. In both subfigures, the top row resembles the LD embedding in 2-D (e.g. $K = 2$) and the bottom row shows the image domain with a continuous colormap based upon the positions of the pixels in the LD embedding.

5-3 Experiment 2: Investigation of cost function and hyperparameter configuration

UMAP comes with more parameters that is reasonable to adjust manually. A complete grid search over the entire parameter space, with only a couple variations per parameter, will take multiple workstations too much time and thus is infeasible. This brings rise to the following question, which is at the heart of this discussion:

How sensitive is UMAP to various parameter configurations? Can we classify whether a produced embedding is considered "better" than another, possibly with a combination of spectral and spatial information?

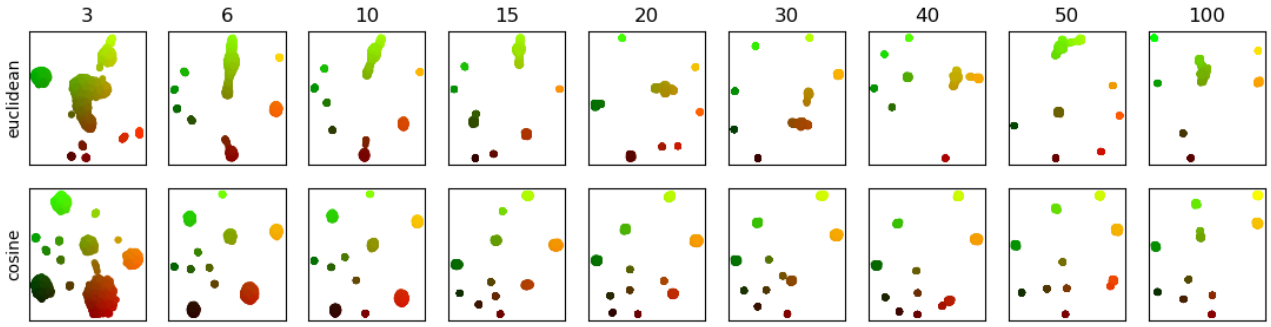
In the quest of an intelligent search through the parameter space, first a semi-grid search is conducted to give a better idea of the sensitivity of the parameter within UMAP. Due to the lack of space, the full results of this search can be found in subsection A-2-1. However, interesting parts of this search will be highlighted during this discussion.

Distance metric evaluation

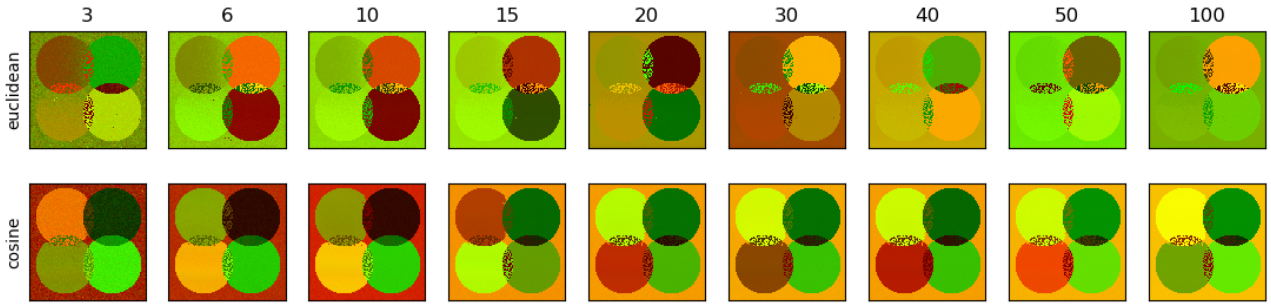
First, we would like to highlight the major differences in choosing the correct distance metric. In combination with the number of neighbors `n_neighbors` for the construction of $\mathcal{G}_{spectral}$, this imposes arguably the largest influence on the LD embedding. For that reason, the results of the semi-grid search for `n_neighbors` have been partly copied to this section for the Euclidean and Cosine distance metrics and can be seen in Figure 5-6. Along with the LD embedding and the image domain in Figure 5-6a and Figure 5-6b, also the evolution of the CE has been plotted in Figure 5-6c. Following these CE plots, the Euclidean distance metric indicates for the spectral as well as the spatial information that the best possible embedding can be found for a low value for `n_neighbors`. However, when observing the LD embedding and the corresponding image domain, the clusters appear more dispersed and the image domain is noisy. The individual circles are more distinguishable with lower values of `n_neighbors` using the Euclidean distance, where for larger values the nonlinear mixing dominates the separation of the clusters.

Moving over to the Cosine distance metric, the optimum of $\text{CE}_{spatial}$ can be found at the lowest possible value for `n_neighbors`. This is intuitive, since $\text{CE}_{spatial}$ penalizes pixel locations and strives for one large 2-D plane with the data points at their acquired positions. Here, the clusters are large and very spread out. This results in some visible noise in the image domain in Figure 5-6b. The minimum of $\text{CE}_{spectral}$ lies more around `n_neighbors` $= 15$ (the default value of UMAP). The four circles remain clearly visible for all tested values of `n_neighbors`, indicating the robustness of the Cosine distance metric.

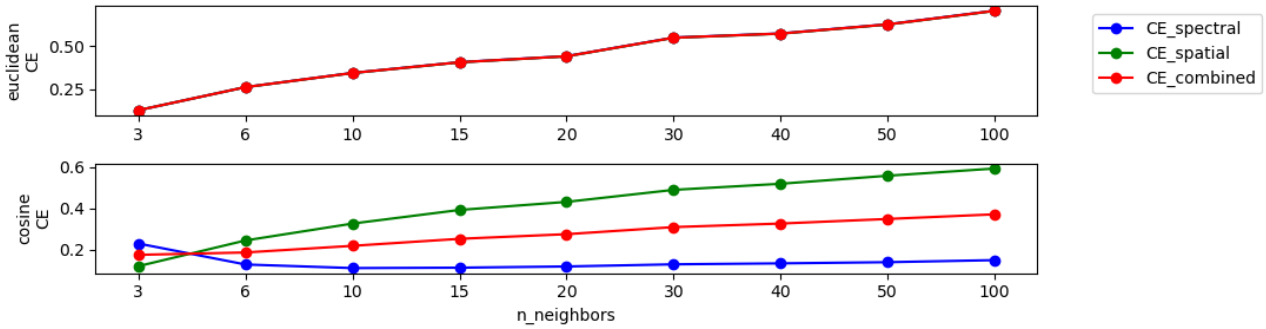
While the parameter $\lambda_{spatial}$ is discussed more in depth during Experiment 3 in section 5-4, we would like to note the importance of the correct distance metric while using said parameter. Taken from Figure A-11, in Figure 5-7 a concise comparison between the Euclidean and Cosine distance metrics has been visualized. Having more spatial information in the mix during the optimization of the Low-Dimensional (LD) embedding raises multiple interesting angles, although here the main focus will lie on the CE plot in Figure 5-7c. This indicates that the optimum for the Euclidean distance lies at $\lambda_{spatial} = 1.0$, by both $\text{CE}_{spatial}$ and



(a) LD embedding



(b) Image domain



(c) CE plot

Figure 5-6: Varying `n_neighbors` for `metric` = *euclidean*, *cosine*. **(a)** LD embedding, **(b)** the corresponding image domain and **(c)** plotting $CE_{spectral}$, $CE_{spatial}$ and $CE_{combined}$.

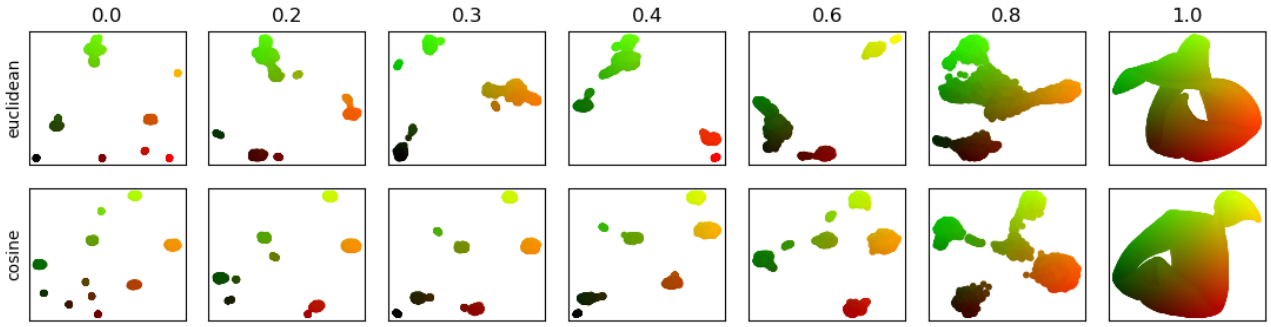
$CE_{spectral}$. Although, when considering the LD embedding and corresponding image domain in Figure 5-7a and Figure 5-7b, we can clearly not distinguish the circles at $\lambda_{spatial} = 1.0$ since all data points are blended into each other. Only the spatial information is used here during optimization. However, even $CE_{spectral}$ implies this embedding to be ideal. Moving to the Cosine distance, we obtain information that is more intuitive. The $CE_{spatial}$ is still lowest at $\lambda_{spatial} = 1.0$ as expected, and $CE_{spectral}$ rises quickly at the same value for $\lambda_{spatial}$. These findings can also be seen in the image domains, since the circles are still very well defined for $\lambda_{spatial} \leq 0.8$. These results show the intuitive and expected responses of using the Cosine distance metric, being robust on the synthetic high-dimensional data used in this experiment.

Other distance metrics While the Euclidean and Cosine distance metric are mostly compared with each other, also the following metrics have been used and can be found in subsection A-2-1: Manhattan, Chebyshev, correlation, Histomatch and Poincaré. It can be seen that the Manhattan, Euclidean and Chebyshev distance all perform similar while executing the semi-grid search. This is to be expected, since these distance metrics are inherently derivatives from the Minkowski distance with different p-values Table 2-2 (Manhattan: $p = 1$, Euclidean: $p = 2$, Chebyshev: $p = \infty$). A relationship between the Cosine and Correlation distance metrics can also be found, since these belong to the so called *angular distance metrics* family. Again, the performance of these metrics is very similar as can be seen in subsection A-2-1.

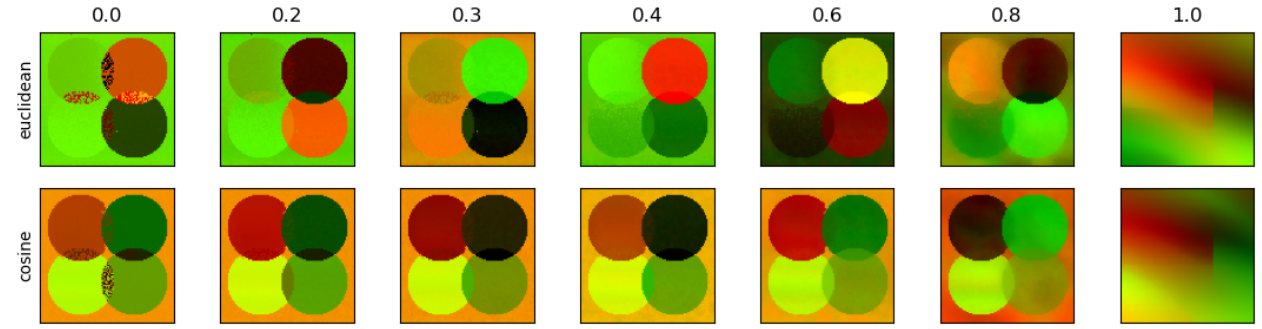
The Histomatch distance is said to perform similarly to the Cosine distance by its authors [78]. However, a stronger relation to the Euclidean distance is found during the semi-grid search indicated by our scoring measurement for $CE_{spectral}$ and $CE_{spatial}$.

Runtime The runtime of UMAP fluctuates under different distance metrics. Here, the number of optimization iterations `n_epochs` will be varied, as well as the number of neighbors `n_neighbors` for the construction of the kNN-graph $\mathcal{G}_{spectral}$. These findings have been summarised in Figure 5-8. All investigated distance metrics except Histomatch scale comparable for increased `n_epochs`. This can be caused by the fact that our implementation of Histomatch is not yet fully optimized for performance (this can be obtained to cache the distance function in machine code). Similar scaling resemblance can be seen when increasing `n_neighbors`. However, more deviation between the distance metrics is visible, with the Cosine distance being able to handle the largest amount of neighbors the quickest.

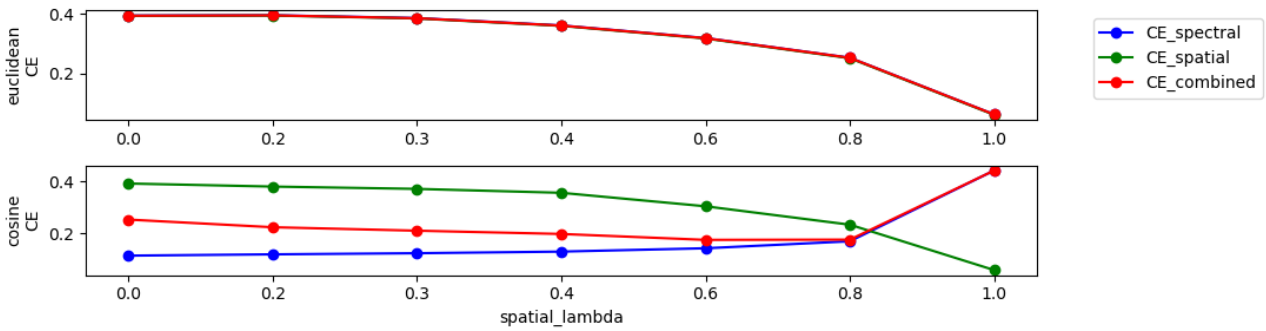
Given the results found in this experiment, the Minkowski-like distances are not fit for evaluating and/or comparing the produced LD embedding since we can not rely on the produced CE. This unfortunately also applies to the Histomatch metric. This leaves us with the angular distance metrics, of which the Cosine distance is being used most extensively throughout the literature, and will thus also be used for subsequent experiments.



(a) LD embedding



(b) Image domain



(c) CE plot

Figure 5-7: Varying `spatial_lambda` for `metric` = *euclidean*, *cosine*. **(a)** LD embedding, **(b)** the corresponding image domain and **(c)** plotting $\text{CE}_{spectral}$, $\text{CE}_{spatial}$ and $\text{CE}_{combined}$.

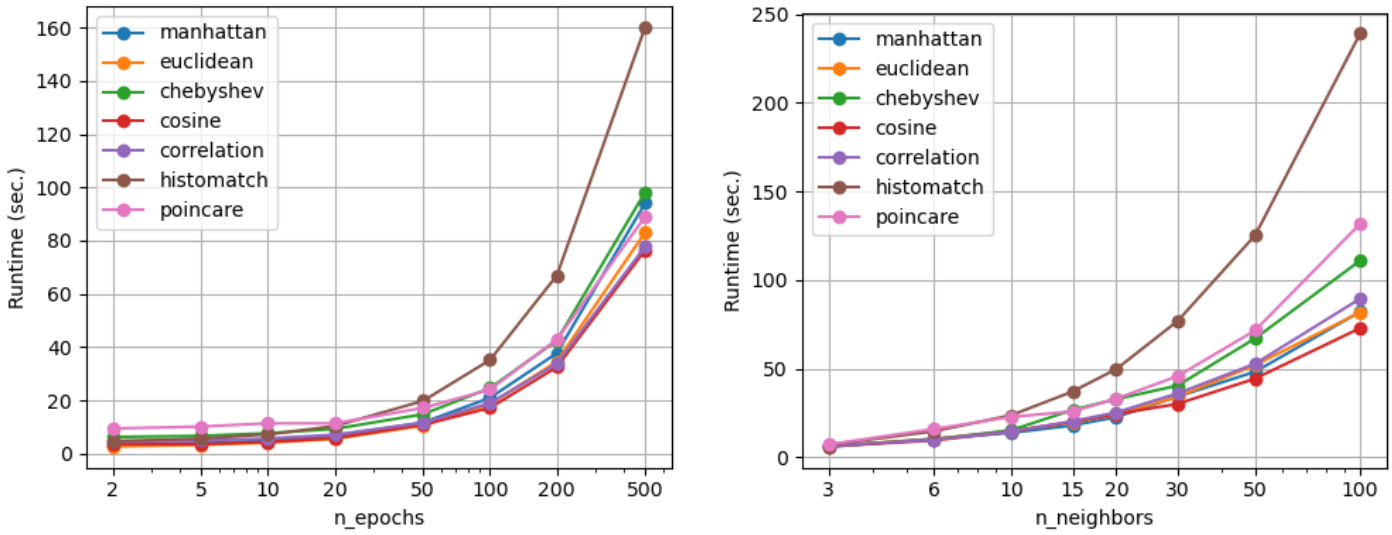


Figure 5-8: Runtime comparison for various distance metrics on the synthetic data set with additive Gaussian noise ($\sigma = 10$) for varying **(a)** `n_epochs` and **(b)** `n_neighbors`.

Mapping sensitive parameters Finally, with the chosen Cosine distance metric, the CE for all varied parameters is plotted in Figure 5-9. Similarly to low values for `n_neighbors` causing large clusters in the LD embedding, large values for `min_dist` establish large clusters as well in the embedding. This is in fact the configuration which is most favourable by $\text{CE}_{spatial}$, as is also indicated in Figure 5-9. While more subtle, increasing `spread` increases the cluster sizes as well. Conjointly, the decline of $\text{CE}_{spatial}$ means an increase in $\text{CE}_{spectral}$ since the clusters become *less defined*.

Large values for parameters such as `negative_sample_size` and `repulsion_strength` appear to be optimal in terms of the evaluation function $\text{CE}_{combined}$. This indicates more optimization iterations at every epoch, and thus more forces are applied to the kNN-graph. In the current scoring scheme, the runtime of UMAP is not taken into account. It remains difficult to add weights to the time UMAP is allowed to optimize its LD embedding, an objective that lies outside the scope of this thesis. A less steep decrease of CE can be seen for `negative_sample_size` and `repulsion_strength` in Figure 5-9. A possible modification could be to monitor the percentage of improvement for increased value of said parameters, and stop increasing when no significant improvement can be obtained. Since these bounds can not be defined right now, we choose not to optimize these parameters.

In Figure 5-9, besides $\text{CE}_{combined}$ also $\text{CE}_{combined,euclidean}$ and $\text{CE}_{combined,manhattan}$ are plotted to indicate the course of these evaluation functions for varying the parameters of UMAP. While all used evaluation functions have similar minima, our implementation in the form of $\text{CE}_{combined}$ is more favourable to the spectral information since too much optimization using only spatial information is undesired and thus we decided to penalize the usage of too much spatial information. The default value for $\lambda_{spatial}$ is set to 0.3, which is the reason for $\text{CE}_{combined}$ and $\text{CE}_{combined,manhattan}$ to not completely overlap for varying `n_neighbors_spatial`. When performing the semi-grid search for the mouse pup data set (subsection; brain), The

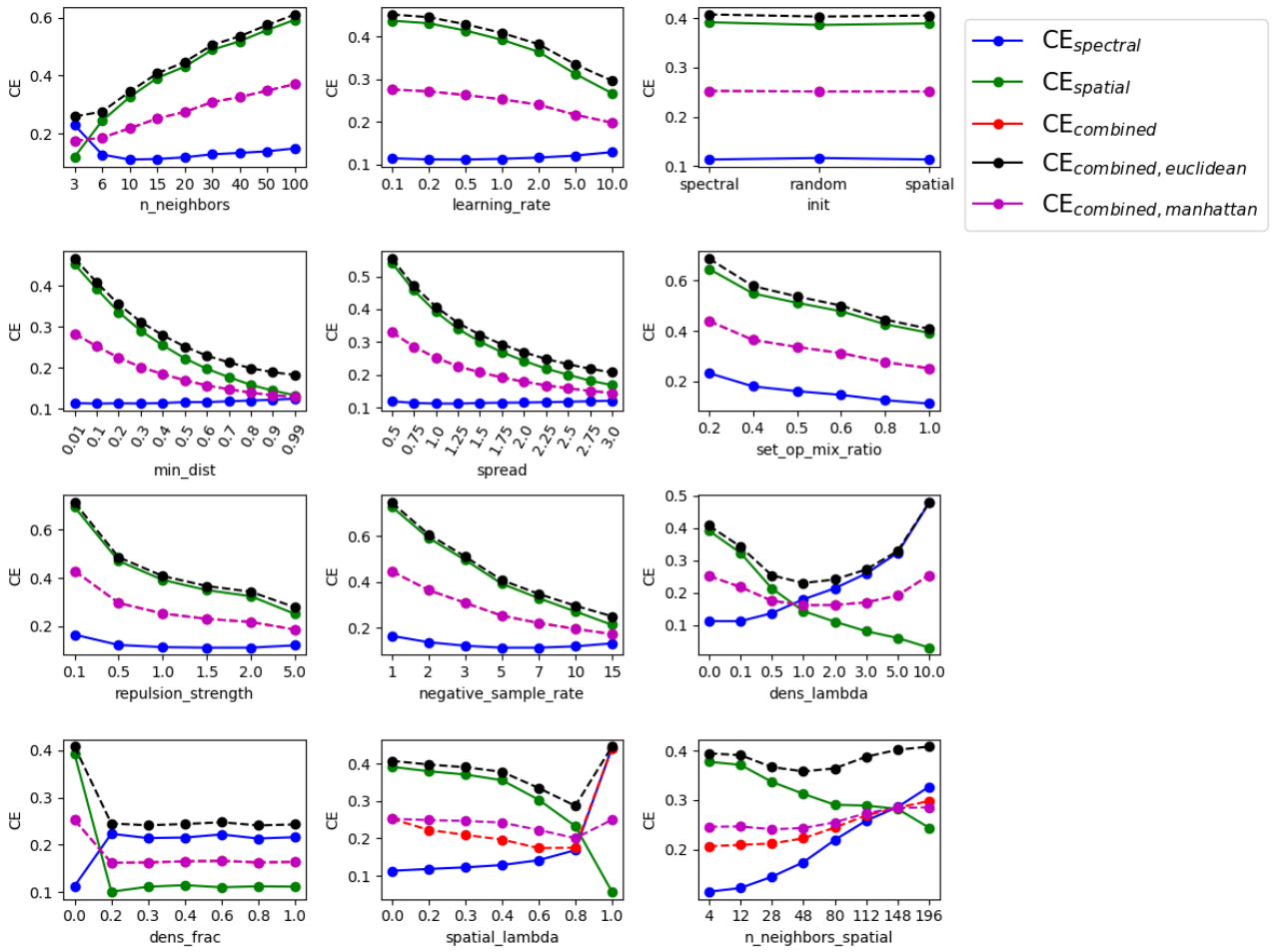


Figure 5-9: Varying numerous parameters of UMAP, and plotting $\text{CE}_{spectral}$, $\text{CE}_{spatial}$, $\text{CE}_{combined}$, $\text{CE}_{combined,euclidean}$ and $\text{CE}_{combined,manhattan}$ (see subsection 3-2-3) for the corresponding parameter configuration. While performing the semi-grid search, all other parameters of UMAP are configured to their default settings, except `metric` $= cosine$, `random_state` $= 100$ and `n_epochs` $= 100$. $\text{CE}_{combined}$ is essentially $\text{CE}_{combined,manhattan}$ for $\lambda = 0.0$, which is the reason why $\text{CE}_{combined,manhattan}$ is plotted over $\text{CE}_{combined}$ for any other parameter evaluation than `spatial_lambda` and `n_neighbors_spatial`.

choice for $\text{CE}_{combined}$ over $\text{CE}_{combined,manahattan}$ is once again motivated in Figure A-14. For varying `spatial_lambda`, $\text{CE}_{combined,manahattan}$ points $\lambda_{spatial} = 1.0$ to be the perfect embedding, this optimization cycle uses only spectral information and completely neglecting spectral information. The resulted image domain is one blur, as will be seen in the upcoming experiment.

The final list of parameters of UMAP, consisting of preset and adjustable parameters and their range, which will be the input for the optimization frameworks in section 5-5 can be seen in Table 5-1.

Parameter (preset)	**Fixed value**
`img_shape`	shape image domain data set *(height* $\times$ *width)*
`spatial_pixel_order`	pixel order of data set $\mathcal{D}$ $(N,)$
`random_state`	42
`n_epochs`	100
`metric`	*"cosine"*
`n_components`	2
Parameter (optimizable)	**Value range**
`n_neighbors`	$[2 - 100]$
`learning_rate`	$[0.1 - 10.0]$
`min_dist`	$[0.0 - 1.0]$
`spread`	$[1.0 - 3.0]$
`set_op_mix_ratio`	$[0.1 - 1.0]$
`spatial_lambda`	$[0.01 - 1.0]$
`n_neighbors_spatial`	$[4, 12, 28, 48, 80, 112, 148, 196]$

Table 5-1: Overview of preset and parameters of UMAP which will be optimized. All unlisted parameters will be configured to their default value. When optimizing `spatial_lambda`, `spatial_flag` will be set to True.

5-4 Experiment 3: Comparing UMAP with UMAPLUS, using spatial information

In subsection 3-3-1 UMAPLUS is introduced, which brings rise to the question how to choose $\lambda_{spatial}$ and `n_neighbors_spatial`. The parameter $\lambda_{spatial} \in [0, 1]$ interpolates between the usage of spectral and spatial information during the optimization of the LD embedding. $\lambda_{spatial} = 1.0$ would be undesirable, as only the pixel locations will be optimized en the spectral information will be completely ignored. We want to investigate whether using a percentage of the spatial information would help the optimization of LD embedding opposed to only using spectral information. It must be noted that the chosen parameters are configured to their default values in section A-1, unless mentioned otherwise. Evaluation of $\lambda_{spatial}$ is visualized in Figure 5-10. Here, the nonlinear mixing in the overlapping parts of the circles is clearly visible with $\lambda_{spatial} = 0.0$, which gets quickly smoothed out for increasing $\lambda_{spatial}$ until only the pixel locations are considered and in $K = 2$ resulting in one 2-D plane.

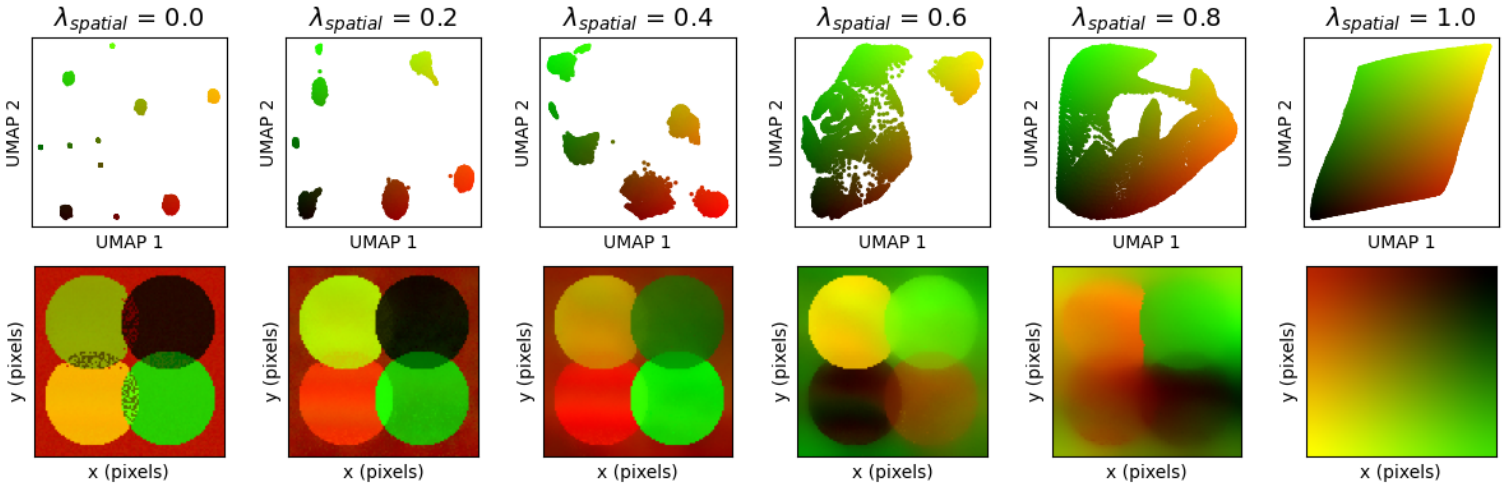


Figure 5-10: LD embedding and image domain with varying $\lambda_{spatial}$ for `n_neighbors_spectral` $= 10$ and `n_neighbors_spectral` $= 28$.

The effect of varying $\lambda_{spatial}$, `n_neighbors_spectral` and `n_neighbors_spatial` is plotted against each other in the form of a heatmap in Figure 5-11, using the spectral component, multiple spatial components and $\text{CE}_{combined}$ as described in Equation 3-16 for both spatial components. It becomes clear that the spectral and spatial components are opposites and favor a small and large value for $\lambda_{spatial}$, respectively. It was noted that the number of computations within UMAP scales with the number of neighbors of the construction of graph $\mathcal{G}$. Increasing `n_neighbors_spectral` thus results in a longer optimization of $\mathcal{G}_{spectral}$, and will be favored for the spectral component in Figure 5-11. Similarly, increasing `n_neighbors_spatial` results in longer optimization of $\mathcal{G}_{spatial}$, and is favored by the spatial components.

For the computation of spatial graph (exact) the CE is similarly constructed as in Table 3-1, using $\sigma_{spatial}$ and $\rho_{spatial}$. It is noticeable that the range while comparing $\text{CE}_{spatial}$ and $\text{CE}_{Moran'sI}$ is different. The difference between both spatial components are the pursuit for conservation of pixel location and the maximum spatial autocorrelation, respectively. Due to the relatively simplistic data set, large areas consist of similar spectral properties (e.g.

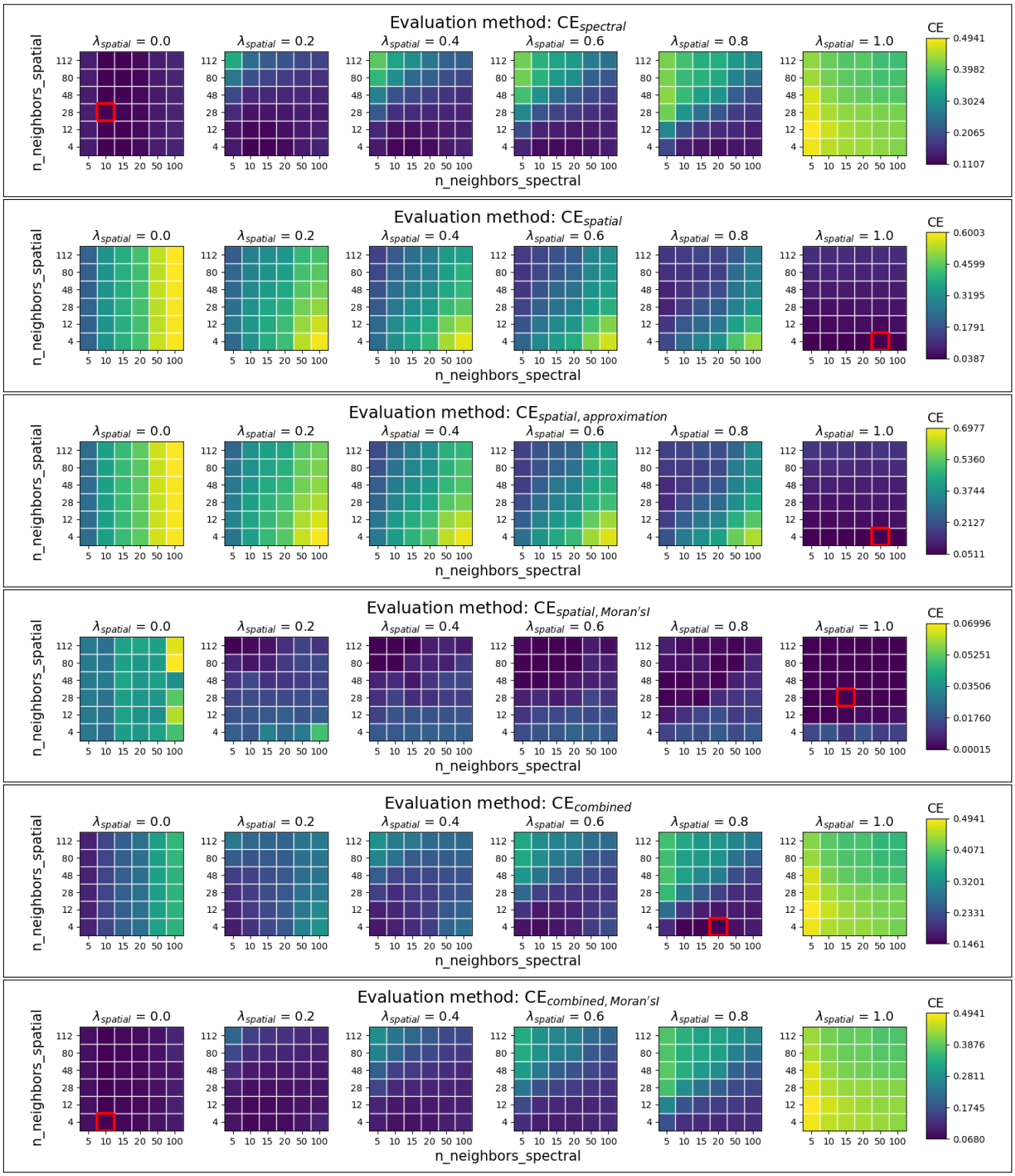


Figure 5-11: Visualization of the spectral component, several spatial components and combination of both components in the form of CE$_{combined}$ for varying $\lambda_{spatial}$, `n_neighbors_spatial` and `n_neighbors_spectral`. For each scoring measure the minimum is highlighted in red. `n_epochs` $= 100$.

the four circles and the background). As previously explained, striving for maximum spatial autocorrelation might not be the best approach when considering the spatial information. The result of combining spatial components with the spectral component can be seen in Figure 5-11, and more details in the minimization of the grid search are recognisable with low values for $\lambda_{spatial}$ using the evaluation method $\text{CE}_{combined}$. The aim for preservation of pixel location over maximum spatial autocorrelation and the broader range of the spatial graph makes it a better option for the evaluation function in Equation 3-16 as $\text{CE}_{spatial,Moran'sI}$.

The obtained minima in Figure 5-11 are highlighted with a red square. The corresponding LD embedding and image domain can be seen in Figure 5-12. The optimal LD embedding pointed out by $\text{CE}_{spectral}$ clearly has the nonlinear mixing present in the image domain. Using the spatial graph evaluation method, this does not construct the 2-D plane as can be seen for the minimum using the Moran's I, but in fact produces strange sharp artifacts in the image domain. The parameters for this minimum are $\lambda_{spatial} = 1.0$, `n_neighbors_spectral` $= 15$ and `n_neighbors_spatial` $= 4$. First of all, it is more difficult for the optimization within UMAP to converge to a perfect 2-D plane due to the initialization, which is configured to `init` = "spectral" in this case. Moreover, the low number of `n_neighbors_spatial` is the more specific reason for the sharp artifacts in the image domain. Since the number of neighbors controls the number of computations within the UMAP algorithm, it can be seen as the *glue* that holds the data points together (e.g. the forces applied on the graph $\mathcal{G}$). When this value is too low, the algorithm optimizes its embedding on a local level, this has also been pointed out in the original paper [11]. We can vary the number of neighbors for the construction of $\mathcal{G}_{spatial}$ and see a better convergence of a 2-D plane in Figure 5-13 for higher values of `n_neighbors_spatial`.

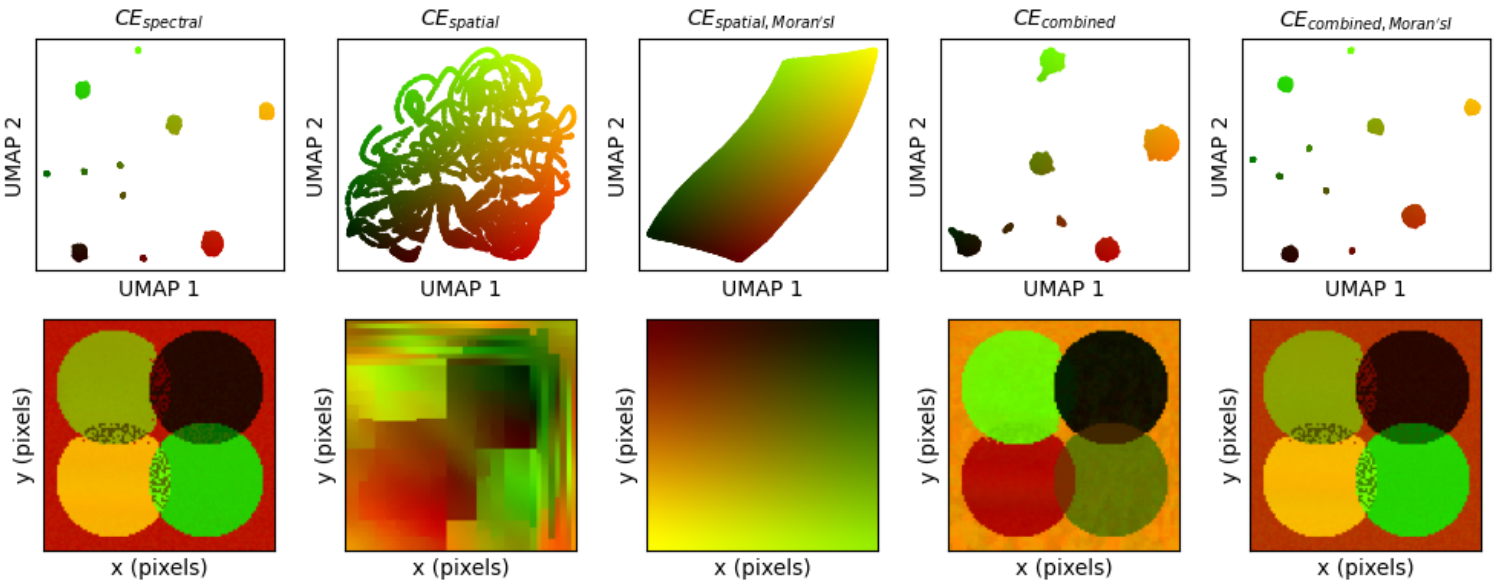


Figure 5-12: LD embedding and image domain of obtained minimum found by grid search using different measures in Figure 5-11. The spatial graph approximation has not been plotted due to the similarity with the exact computation of spatial graph.

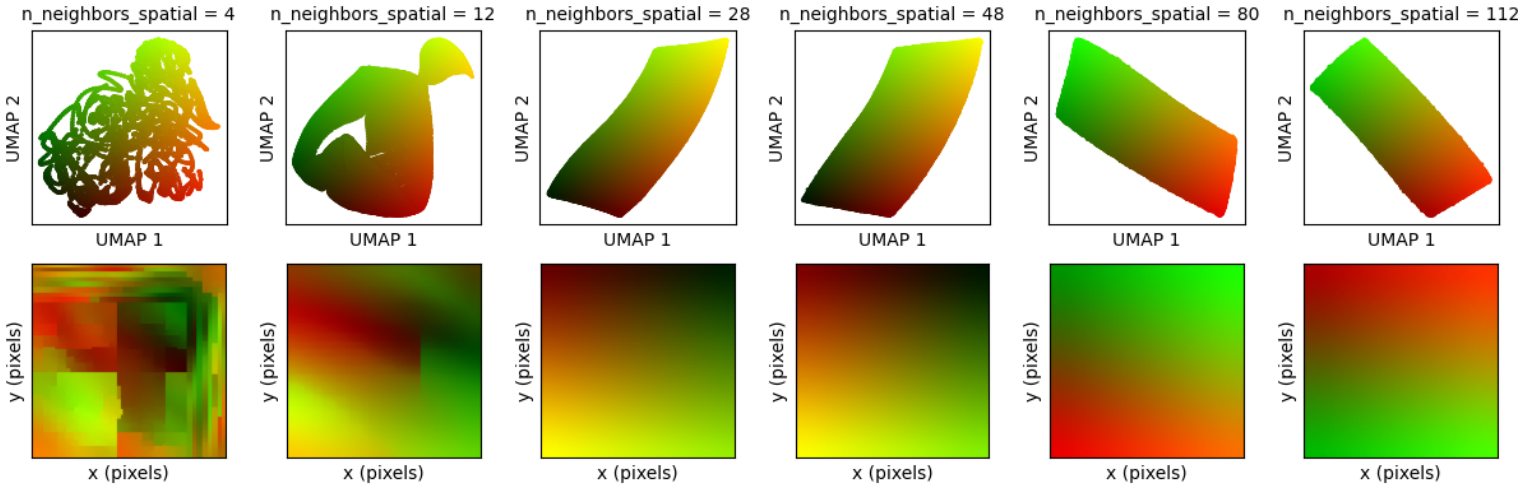


Figure 5-13: Varying the number of neighbors for the construction of the kNN-graph $\mathcal{G}_{spatial}$ in order to show the increment in forces converging to a 2-D plane for higher values of `n_neighbors_spatial`. `n_epochs` $= 100$.

The convergence to a 2-D plane of $\mathcal{G}_{spatial}$ can also be seen when we plot this graph within the LD embedding in Figure 5-14. For `n_neighbors_spatial` $= 12$, a large part of the pixels in the LD embedding are positioned at the pixel locations relative to the neighboring pixels. However, we can see the structure of the graph "tearing apart" in the middle, indicating not enough force is applied during optimization of the LD embedding. For higher values of `n_neighbors_spatial`, the 2-D plane is fully connected and focuses more on the orientation of the image domain (e.g. top and bottom pixels in the LD embedding are the top and bottom pixels in the image domain). It must be noted that spectral initialization is used, choosing initial positions based upon the spectral information.

We can conclude that $\text{CE}_{combined}$ is capable to filter the nonlinear mixing in the synthetic data set from the image domain by increasing $\lambda_{spatial}$. Caution must be taken not to increase $\lambda_{spatial}$ too much, since only spatial information is used when $\lambda_{spatial} = 1.0$, which is undesirable. Increasing `n_neighbors_spatial` effectively brings neighboring pixels in the image domain closer to each other in the LD embedding.

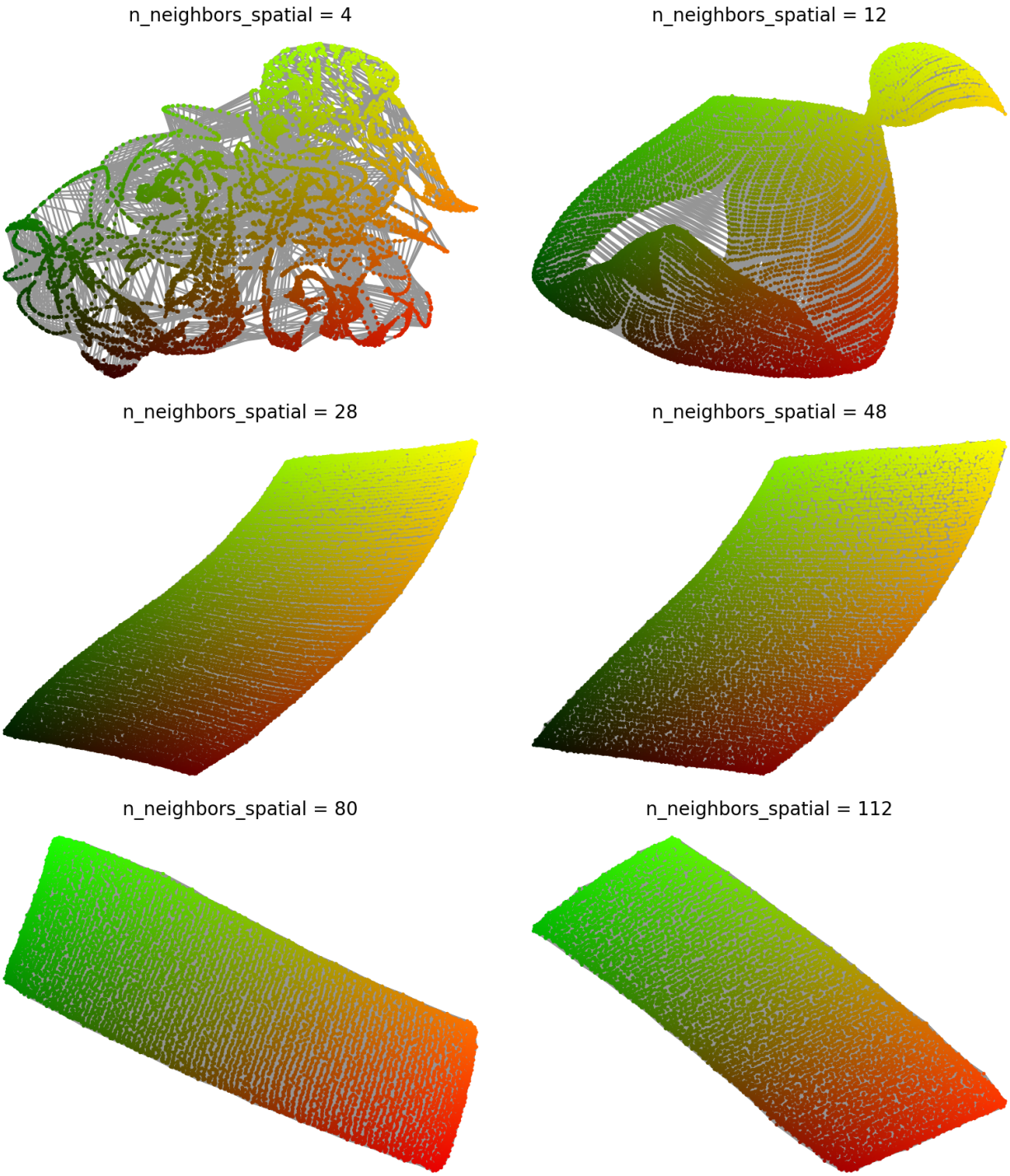


Figure 5-14: Visualization of $\mathcal{G}_{spatial}$ for varying `n_neighbors_spatial`. The visualized connections are between the k nearest neighbors in the image domain. More nearest neighbors causes more force to applied for optimizing the LD using (spatial) information, and thus a 2-D array is constructed. `n_epochs` $= 100$, $\lambda_{spatial} = 1.0$.

5-5 Experiment 4: Automation searching parameter space (DD-UMAP)

Finally, this section will apply the proposed optimization frameworks described in section 3-4 on the synthetic and real-world IMS data sets in order to optimize the parameters by UMAP. The research question that is at the center of this experiment is:

Is Data-Driven UMAP (DD-UMAP) able to find the optimal hyperparameters and thus produce the best possible embedding? What optimization method (1-D or Bayesian optimization approach) is more suitable to find the global optimum in the defined parameter search-space? And how does DD-UMAP compare to traditional grid search?.

The generated synthetic data set with additive noise ($\sigma = 10$) is used throughout this experiment.

1-D optimization

The 1-D optimization utilizes the Golden section search to find the optimum parameter configuration. Taking this approach, the parameters of UMAP can only be optimized in an sequential manner (e.g., one parameter at a time).

Small framework

To see how the algorithm finds its optimum, only `spatial_lambda` is being optimized with range $[0.0, 1.0]$. The first 8 iterations of the search have been plotted in Figure 5-15. The input configuration of the first 2 iterations are determined prior, based on the golden section ratio. From iteration 3, the algorithm is started and, as can be seen, the minimum of the underlying cost function can be found. The true function is unknown in practice, but in this case is achieved through an extensive grid search with 100 iterations.

Plotting the x-axis values ($\lambda_{spatial}$) and the y-axis values (CE) separately, we can visualize the convergence of the CE and parameter value in Figure 5-16a and Figure 5-16b. Every iterations takes 100 epochs, so over time this has been depicted with the cumulative number of epochs by UMAP.

Two types of stopping conditions are monitoring the Golden section search. The algorithm will be terminated when either the parameter change ($\theta_{param} = 3\%$) or change in CE ($\theta_{CE} = 5\%$) is within a percentage bound, and these values stay within these bounds for a number of iterations ($\Delta_{theta} = 3$). This active observation is visualized in Figure 5-16c. In this case, the CE change is within the given limits given prior to the algorithm, thus the reason for terminating any further iterations. In this case, the algorithm is terminated based upon CE change. An adequate value for $\lambda_{spatial}$ is obtained, with an error of $\frac{0.711}{0.7} \times 100 \approx 1.6\%$.

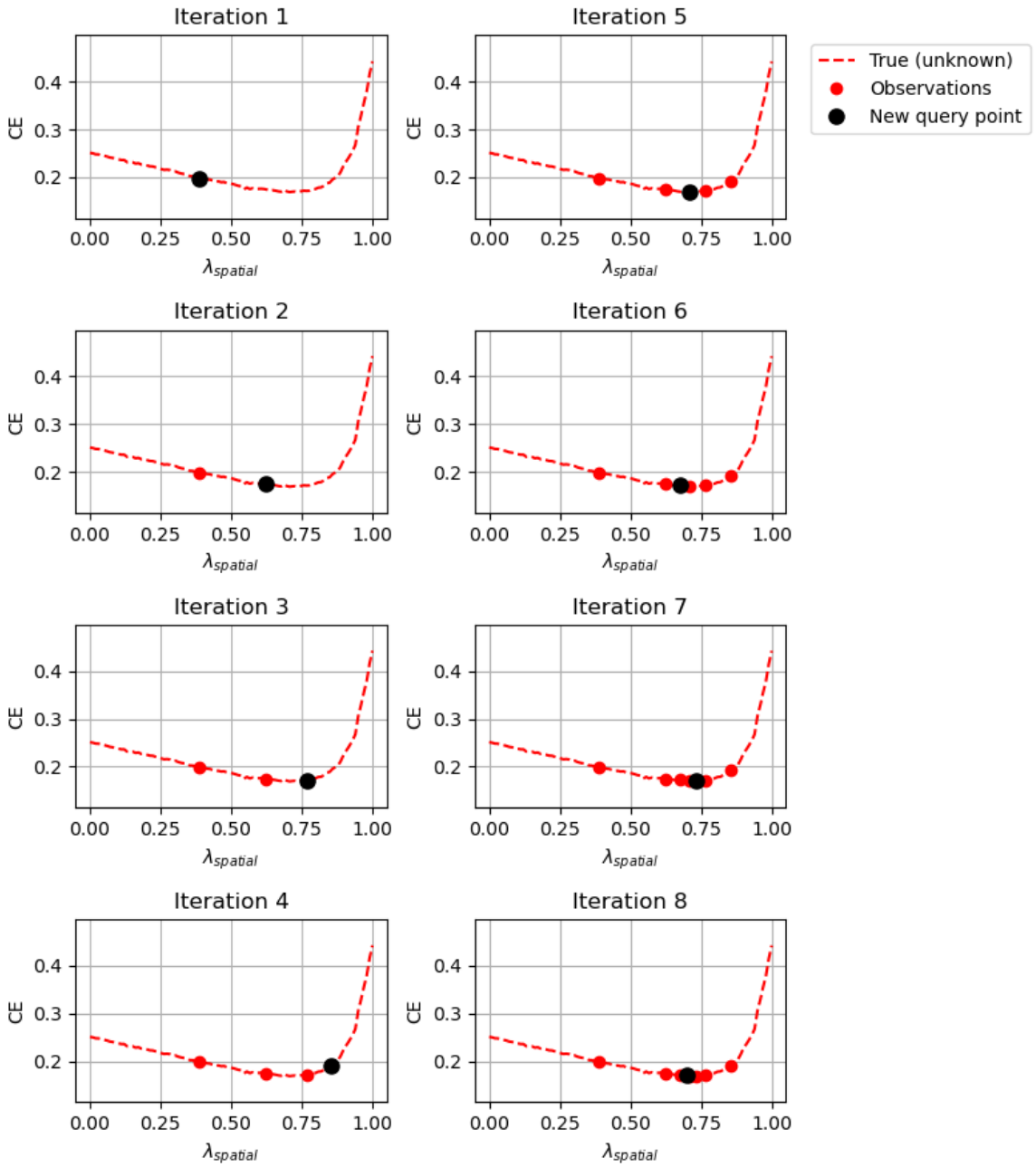
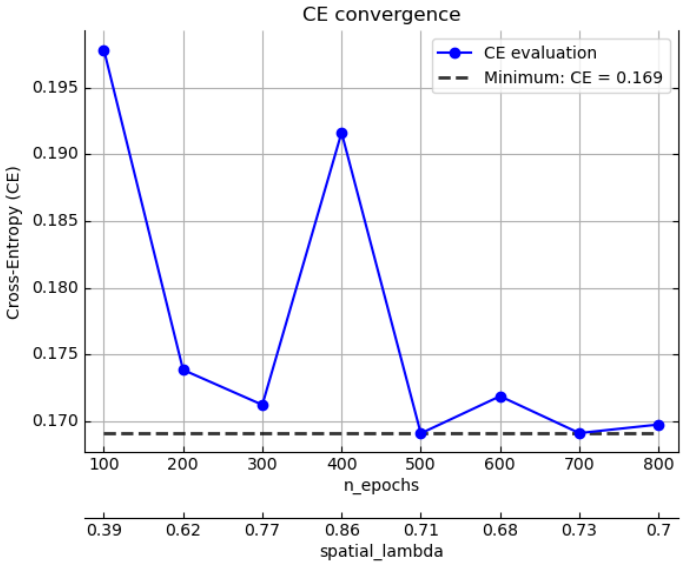
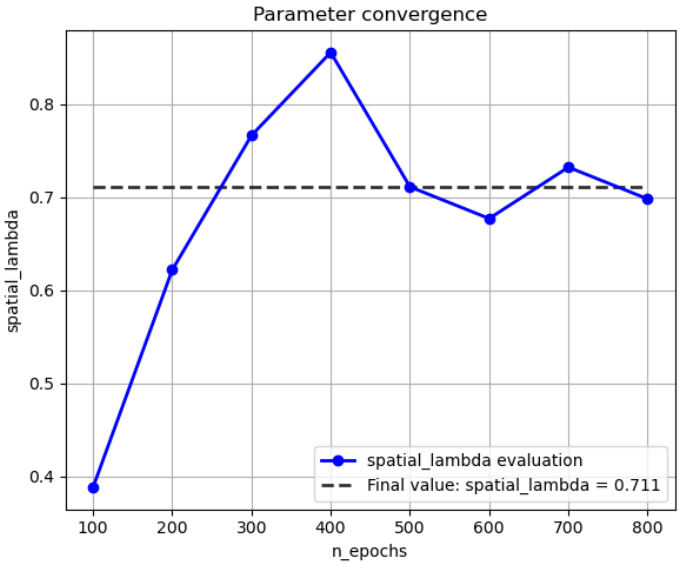


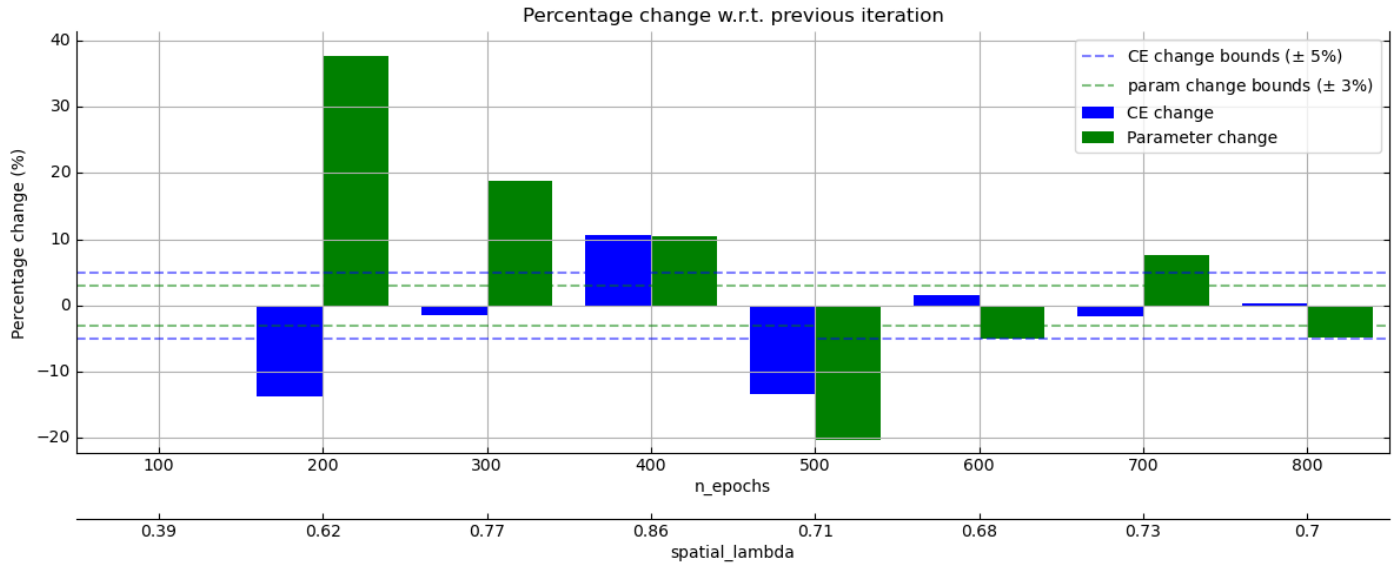
Figure 5-15: The first 8 iterations of the Golden section search algorithm, optimizing parameter `spatial_lambda` on the synthetic data set with additive Gaussian noise ($\sigma = 10$). The true function is plotted in red striped and is unknown prior to the start of the algorithm.



(a) CE convergence



(b) Parameter convergence



(c) Percentage change evolution

Figure 5-16: Visualization of the CE and parameter convergence executed by the Golden section search, optimizing $\lambda_{spatial}$.

Medium framework

Next, multiple parameters will be considered for this optimization. We can restart from the default settings of UMAP with every parameter switch, and this would result in finding the minimum in the function approximated with the semi-grid search in Figure 5-9. However, saving the optimum value for the optimized parameter and use these values for subsequent parameter searches is more interesting. In this case, the order may be important since not the complete parameter space will be explored and only one parameter is varied at a time (and the values of all other parameters are locked into place). We will optimize `spatial_lambda`, `n_neighbors` and `n_neighbors_spatial`. The order will be different, thus creating 6 possible combinations in order to investigate whether the order in which the parameters are optimized does matter. The findings of all framework using the 1-D optimization can be found in Table A-4.

In the independent case, the parameters are not set to their optimized value but reset to their default settings. While most end-values for $\text{CE}_{combined}$ are similar, outliers do exist for certain parameter orders. the parameters `spatial_lambda` is highly influential for every other parameter it will be optimized with, since $\lambda_{spatial}$ is integrated in the evaluation function $\text{CE}_{combined}$. With these settings, `n_neighbors` and `n_neighbors_spatial` achieve consistently 4 and 12, respectively. Apparently the value for $\lambda_{spatial}$ has its optimum either at 0.333 or 0.711. The independent case achieves a similar parameter configuration as the cases with different orders, although $\text{CE}_{combined}$ is larger. Since the computed minima of the parameters are not taken into account for subsequent optimization cycles, we restrict parts of combinations which could lead to a better optimum. Neither approach is ideal, since for the independent case we rely on default values of the parameters close to the optimum to achieve an LD embedding close to the global minimum. It remains difficult to choose the correct order beforehand. And even if we could, a large part of the parameter space will be restricted since we are optimizing parameters one at a time while fixating the remaining parameters.

Caution must be taken when the underlying function has multiple local minima. This is not the case during the semi-grid search on the synthetic data with additive Gaussian noise ($\sigma = 10$) in Figure 5-9, but this can be observed in Figure A-14. Here, a semi-grid search using a subsection of the brain of the mouse pup data set for varying `n_neighbors_spatial` shows a local minimum for $\text{CE}_{combined}$ around `n_neighbors_spatial` $= 112$. A method such as 1-d optimization using Golden section search could potentially be stuck at this minimum, since the algorithm has no exploration term found in the acquisition function α belonging to the Bayesian optimization. The global minimum at `n_neighbors_spatial` $= 4$ will could potentially not be found. A multi-start Golden section search or changing the outer bounds of the search-interval can neutralize this risk.

Large framework

In the large framework, all parameters mentioned in Table 4-2 will be considered. As discussed in the previous subsection, the order of optimizing the parameters influences the obtained $\text{CE}_{combined}$. Checking all possible order combinations for 7 parameters would lead to $7! = 5,040$ algorithm starts. This is simply infeasible, so we will only check the independent case again, and several random cases. These findings have been summarized in Table A-4. Interestingly, the parameter values stay relatively stable, independently from the order in

which they are optimized. The value for `spatial_lambda` is low for most parameter orders. A value for $\lambda_{spatial}$ close to zero means less impact by optimization of $\mathcal{G}_{spatial}$, rendering the choice for `n_neighbors_spatial` negligible, while a high value for `n_neighbors_spatial` does increase the runtime substantially. Such runtime-constraint can potentially be taken into account in future work.

Random search

Secondly, a random search is performed due to its simplicity and possibility to be performed in parallel. Random search also relinquish the execution of going through every possible configuration (grid search), avoiding long runtimes. However, one can not be certain that the optimum is found after only one executed run. Therefore, multiple runs are performed, with the best possible combination using the small, medium and large framework all bundled together in Table A-5.

Bayesian optimization

Lastly the Bayesian framework is applied to our parameter optimization problem. A similar division between a small, medium and large framework will be discussed.

Small framework

Similar to the Golden section search, the small framework only optimizes one parameter: $\lambda_{spatial}$. We start the algorithm with 2 random searches, and perform 5 iterations following the Bayesian rules. In total 7 iterations have been plotted in Figure 5-17. It can be seen that the maximum value for the acquisition function determines which input parameter configuration (e.g. value for $\lambda_{spatial}$) will be used for the next iteration. The acquisition function trades between exploitation and exploration. The uncertainty of the underlying unknown function for $\lambda_{spatial}$ is indicated with a green-filled region, officially noted as the Gaussian Processes (GP). Larger bands of the GP depict larger uncertainty of that region and thus will lead to high values for the acquisition function, making this region interesting for further investigation. Iteration 5 and 6 show the exploitation: areas with large uncertainties are being sampled. When most uncertainties are lifted, iteration 7 shows exploration: investigation of areas that are yet unknown. Since the true function can be seen in Figure 5-17, we know that $\lambda_{spatial} = 1.0$ is infeasible. This will be seen by the Bayesian optimization method as well starting from iteration 8, moving quickly to the region around $\lambda_{spatial} \approx 0.7$. All together, a good approximation of the underlying true function for $\lambda_{spatial}$ can be found in only 7 iterations (3 random and 4 intelligent searches). Multiple runs show that most obtained values for `spatial_lambda` stay within a few percent from the (through an intensive grid search obtained) true minimum.

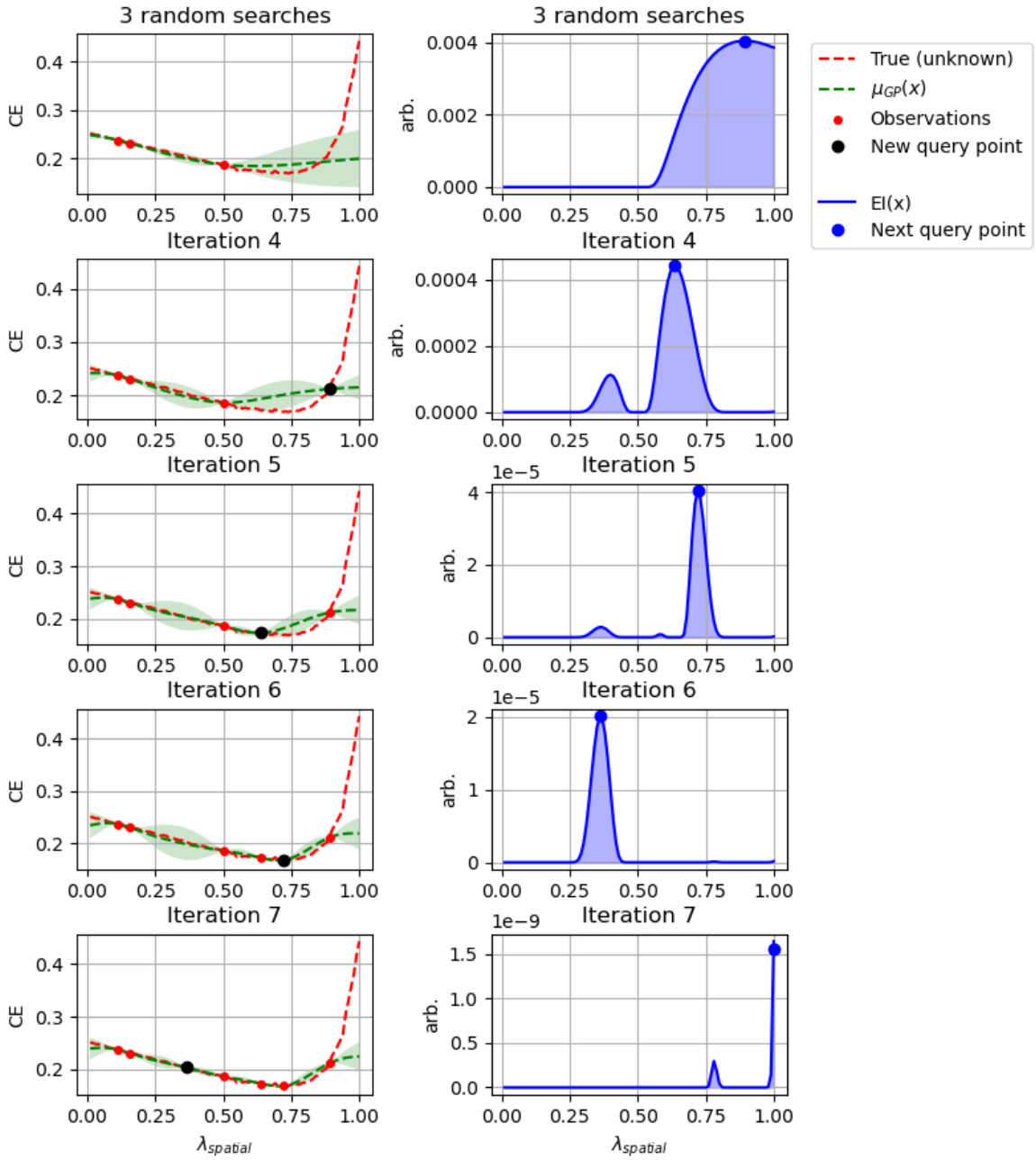


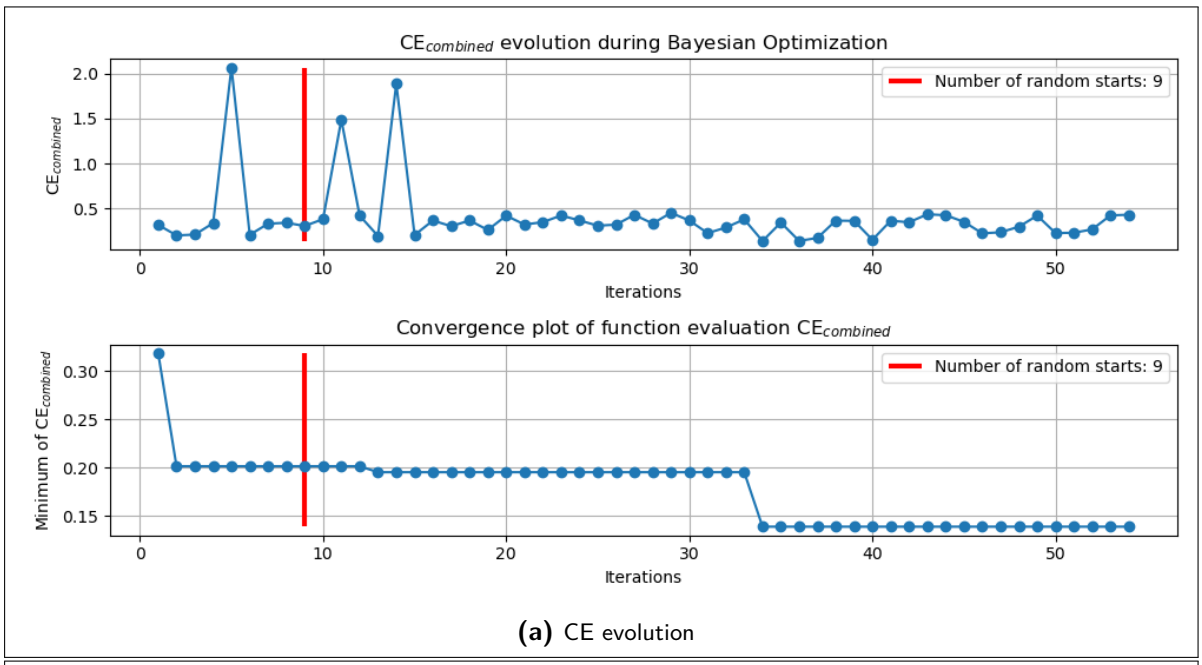
Figure 5-17: Visualization of the first 7 iterations of optimizing $\lambda_{spatial}$ following a Bayesian framework. The 7 iterations consist of 2 random searches and 5 intelligent searches by maximizing the acquisition function α. The underlying true function for $\lambda_{spatial}$ is plotted in red-striped and is unknown in practice.

Medium framework

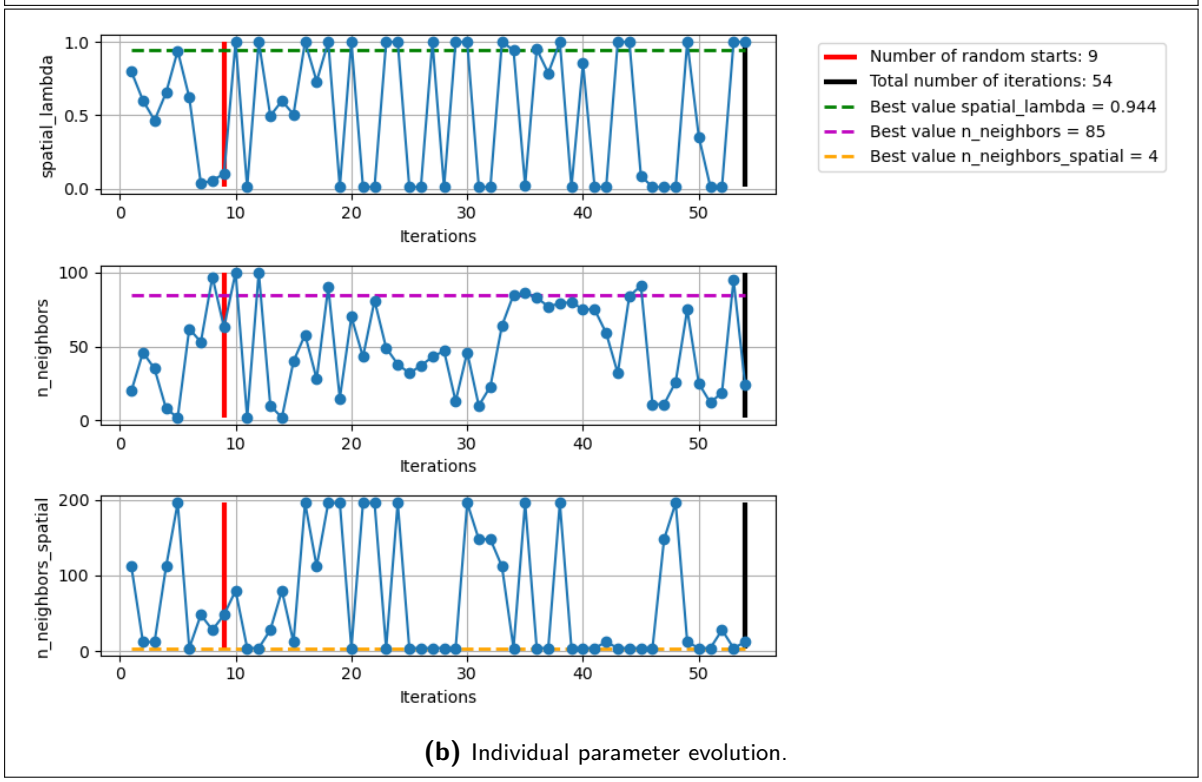
For a well defined comparison between the previous parameter optimization methods, the parameters `spatial_lambda`, `n_neighbors`, and `n_neighbors_spatial` will be investigated here as well. The evolution of $CE_{combined}$ and the parameter configurations throughout the optimization phase can be seen in Figure 5-18. The number of random starts are highlighted by a red bar, after which the Bayesian optimization framework is determining the next optimal parameter configuration based upon the maximization of the acquisition function α. Introducing a threshold is discussed but not applied during this experiment. Prior to the parameter search it is unknown how many iterations are needed. No threshold is applied, which could potentially terminate the algorithm before it has had the chance of exploring the parameter space thoroughly. During the individual parameter evolution in Figure 5-18, once a better LD is found at iteration 34, the `n_neighbors` stays at this value while the algorithm is tuning the other buttons (`spatial_lambda` and `n_neighbors_spatial`). After a preset number of iterations, unfortunately no further improvement can be found.

Large framework

In the large framework, all parameters mentioned in Table 4-2 will be considered. Similarly to the 1-D optimization and the random search, the minimum obtained value is lower when adding more parameters to the optimization procedure. Some parameters vary more than others for different optimum parameter configurations. Parameters such as `min_dist` and `spread` and `set_op_mix_ratio` stay relatively consistent around a certain value, even for multiple runs. In the semi-grid search executed in subsection 4-2-3 (specifically Figure 5-9), we have seen that these parameters have their minima at the same values found by the Bayesian optimization approach. This, in combination with the constant values found by the 1-D optimization approach gives rise to the conclusion that the parameters are not as multivariately interconnected as initially assumed.



(a) CE evolution



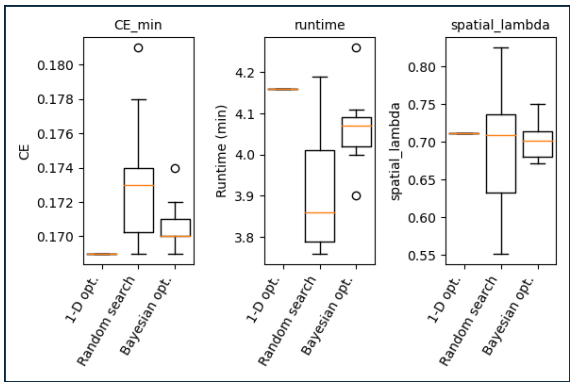
(b) Individual parameter evolution.

Figure 5-18: Bayesian optimization applied to the synthetic data set with additive Gaussian noise ($\sigma = 10$), optimizing `spatial_lambda`, `n_neighbors`, and `n_neighbors_spatial`. **(a)** CE$_{combined}$ is plotted at every iteration at the top subplot, while the lower subplot show the minimum CE$_{combined}$ over all previous iterations. **(b)** The input configuration . This specific run has 9 random searches, and 15 intelligent searches following the Bayesian optimization framework through maximization of the acquisition function α, leading up to 24 total iterations.

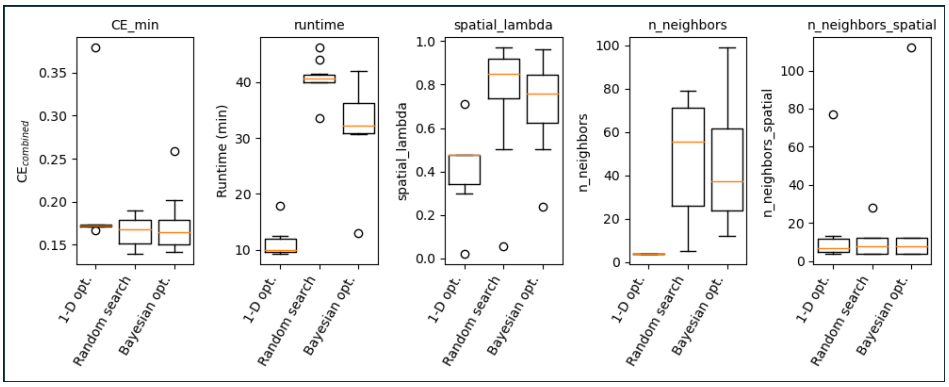
Final remarks

All information of this experiment (the variation in the minimum obtained value for $CE_{combined}$, elapsed runtime and optimal parameter configuration) is combined in Figure 5-19. Interestingly, the 1-D optimization method is outperforming both random search and the Bayesian approach for the small and large framework. In terms of runtime, all methods perform similarly for the small framework, but when scaling to 3 or 7 parameters the difference becomes substantial. It must be noted that the random search and Bayesian optimization are allowed to take more iterations, although the biggest difference comes from exploring high values of `n_neighbors` and `n_neighbors_spatial`, leading to considerable average runtimes. The 1-D optimization does find optimal values for `n_neighbors` and `n_neighbors_spatial` at low values, and uses partly the default values for these parameters, which are also relatively low (see section A-1). Once the optimal value for this parameter (pointed to by the evaluation function) would be higher, the difference in runtime would be smaller between different optimization methods. The expectation was that a multivariate approach such as Bayesian optimization would perform equal to or better than the 1-D optimization method. The fact that the opposite is true could be explained by the fact that the Bayesian optimization did not receive enough iteration steps to explore the vast parameter space. Large spaces will not necessarily result in much improvement, yet these areas will also be partly explored due to the exploration property of the acquisition function. Large deviation in the values for the large framework using the Bayesian optimization can indicate that more iterations are necessary to find the global minimum (see Table A-6). The 1-D optimization using the Golden section search merely focuses on obtaining the minimum value. Also, the parameters are perhaps not as multivariately interconnected as initially assumed.

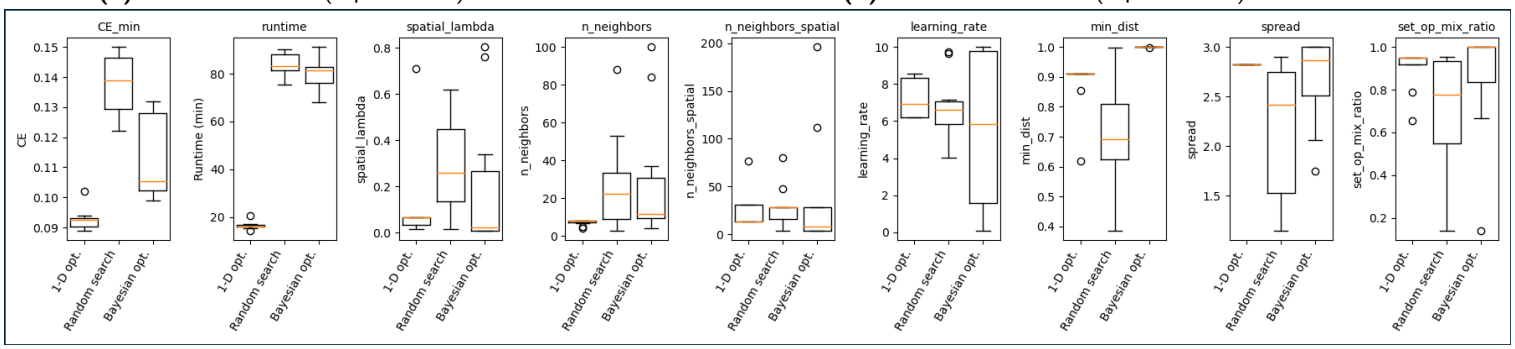
The final optimized LD embeddings found by each optimization method for varying framework size can be seen in Figure 5-20. the default settings shows the nonlinear mixing, and will be resolved in basically every framework, except for DD-UMAP using the 1-D optimization (Figure 5-20f) and the Bayesian optimization (Figure 5-20h) using the large framework. Apparently, a large value for $\lambda_{spatial}$ is less beneficial given the available parameters left for optimization. During the semi-grid search in experiment 2, parameters such as `min_dist`, `spread` and `n_neighbors` have different values than used in the embedding in Figure 5-20f and Figure 5-20h.



(a) Small framework (1 parameter)

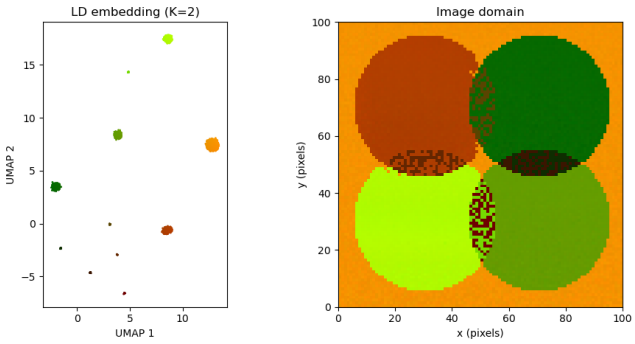


(b) Medium framework (3 parameters)

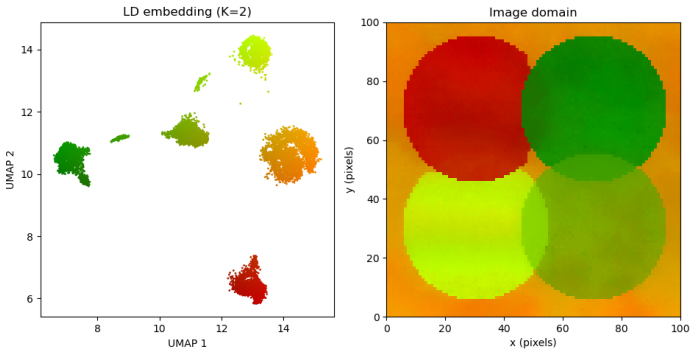


(c) Large framework (7 parameters)

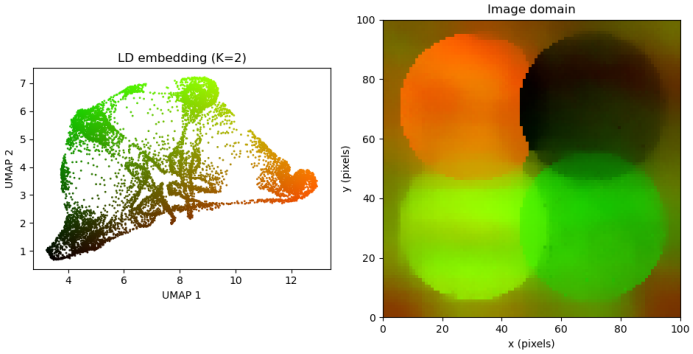
Figure 5-19: Comparison of the three types of optimization methods: 1-D optimization using Golden section search, random search and Bayesian optimization. Three frameworks are visualized; **(a)** small framework with 1 parameter, **(b)** medium framework with 3 parameters and **(c)** large framework with 7 parameters. The information contains the best obtained value for $CE_{combined}$ (denoted as CE_{min}), the runtime it took to execute the search/optimization and finally the variation of the optimal parameter configuration, based on multiple runs of the mentioned optimization methods. The values behind these figures can be found in Table A-4, Table A-5 and Table A-6.



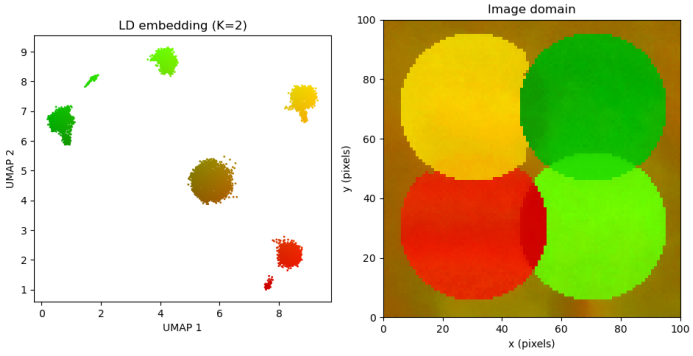
(a) UMAP with default settings



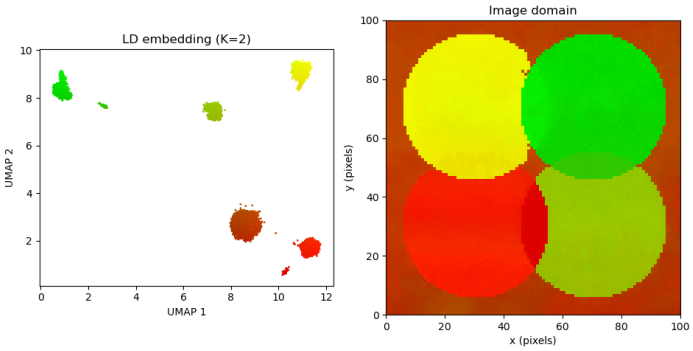
(b) DD-UMAP, small framework



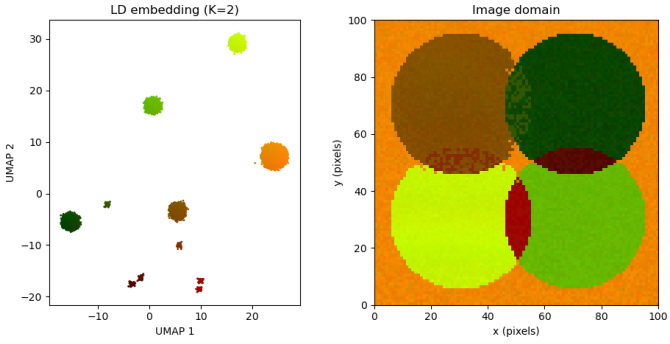
(c) DD-UMAP, 1-D optimization, medium framework



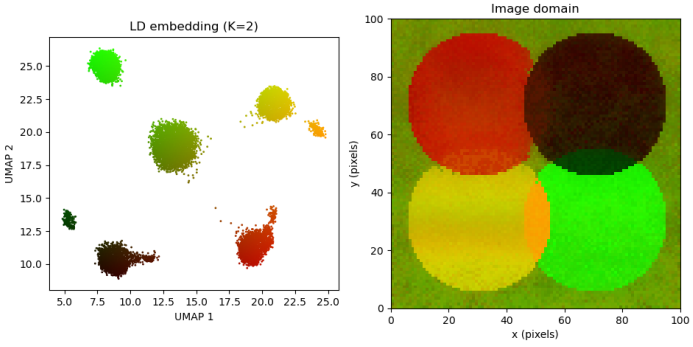
(d) DD-UMAP, random search, medium framework



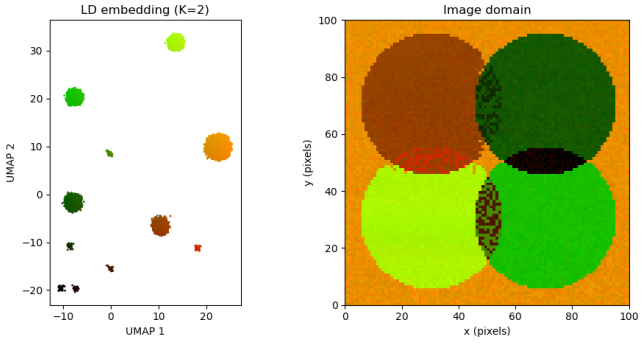
(e) DD-UMAP, Bayesian optimization, medium framework



(f) DD-UMAP, 1-D optimization, large framework



(g) DD-UMAP, random search optimization, large framework



(h) DD-UMAP, Bayesian optimization, large framework

Figure 5-20: Optimized LD embeddings. Produced from the minimum $CE_{combined}$ found over multiple runs (see section A-3).

5-6 Case study: Assessment on real-world IMS data sets

In this case study, two real-world IMS data sets are subjected to UMAPLUS and DD-UMAP: a mouse pup and murine kidney data set. In Experiment 0 in subsection 4-2-1, we have motivated the subsampling for the full mouse pup data in order to evaluate the LD embedding. The results for the computation of the confidence score ϕ on the data sets used in this section can be seen in Table 5-2. Interestingly, less data points are necessary for the kidney data set compared to the mouse pup data set. This can be explained due to the less prominent structure. In Figure 5-21 and Figure 5-22, the difference in structure in the data is clearly visible; the mouse brain has clear separation, whereas the subsection of the kidney shows a homogeneous pattern. More details within each data set between the different dimensionality reduction techniques is discussed in separate subsections. When applying DD-UMAP, the large optimization framework is considered, optimizing 7 parameters mentioned in Table 4-2. Some LD embeddings produced by DD-UMAP are transformed such that the resulted colormap is sufficiently similar to regular-UMAP, making comparison possible.

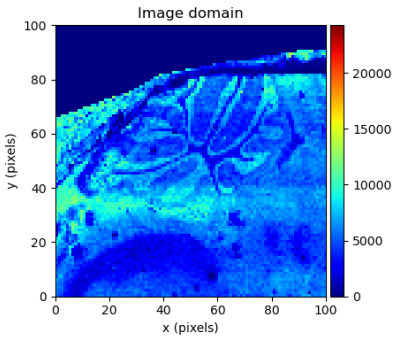
Data	N	**Percentage used for evaluation (%)**	N_{subset}	**Confidence score ϕ (%) (lower is better)**
Mouse pup, brain	8174	10	817	0.380
Mouse pup, full	164808	1.3	2142	0.787
Kidney, outside layer	10000	1	100	0.832
Kidney, right-half side	168886	0.1	168	0.847

Table 5-2: Overview of the percentage used for evaluation by $\text{CE}_{combined}$ with the corresponding real-world data set. In order to use a certain percentage of the data, we state that the confidence score ϕ must be below 1%. DD-UMAP is not applied to the full mouse pup, but the confidence score is computed to highlighted the scaling of the necessary percentage of the data set for evaluation by $\text{CE}_{combined}$ between the mouse pup and the murine kidney.

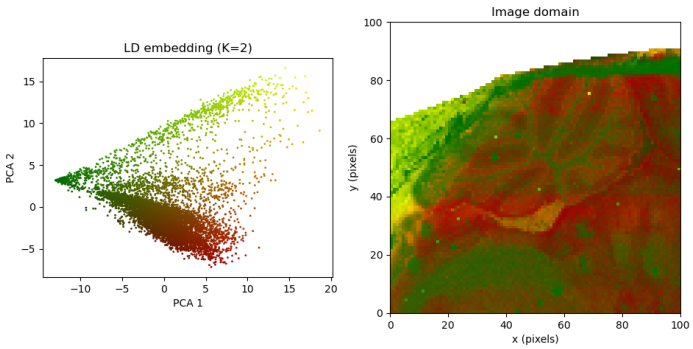
Mouse pup data set (subsection; brain)

First, a subsection from the mouse pup data set containing the brain is considered. In Figure 5-21, first an ion image with m/z value = 836.634 is shown. With only one ion image, some structure can be seen already. A single ion image tells us much about a certain m/z value, but not about the total chemical consistency across the entire spectral domain. A PCA decomposition with 2 target dimensions shows already 72.95% of explained variance. The resulting LD embedding does not contain well-defined clusters, showing a vague structure in the image domain, but not clearly visible. The LD embedding produced by t-SNE and UMAP are adequately similar, with the UMAP embedding producing even better defined clusters (more compact). Two types of optimization methods (1-D optimization with Golden section search and a Bayesian approach) are compared with spectral and spatial initialization. Noteworthy is the relatively larger cluster size in the LD embedding for all versions of DD-UMAP, especially when using spatial initialization. This potentially can be explained due to the part of the evaluation function $\text{CE}_{combined}$ that favors not only spectral, but also spatial information, causing the data points in the LD embedding to spread out more evenly. Larger cluster size results in less defined structure in the image domain, as can be seen in the PCA decomposition in Figure 5-21b.

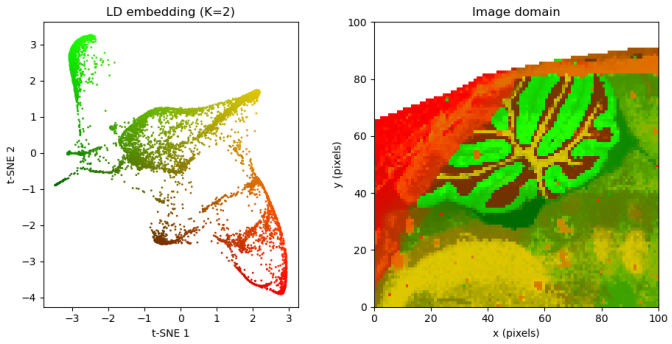
Not all parameter configurations produced by DD-UMAP are plotted here, but 1-D optimization chooses a low value for `n_neighbors` (4), whereas the Bayesian approach finds an optimum at `n_neighbors` = 14. A lower value for `n_neighbors` in dispersed clusters, see Figure 5-21. Also, finer structure becomes visible in the left side of the image domain for lower values of `n_neighbors`, which is not visible in the embedding produced by UMAP using default settings. Overall, higher contrast is created by DD-UMAP in the image domain, differentiating the different regions more strongly than regular-UMAP. DD-UMAP finds even more structure between the trenches of the cerebellum. In regular-UMAP, two spots in the down left corner of the image domain are barely visible, since the spots are green against a green background. DD-UMAP does detect these spots (best visible in Figure 5-21f), thus indicating dissimilarity of these spots compared to the background.



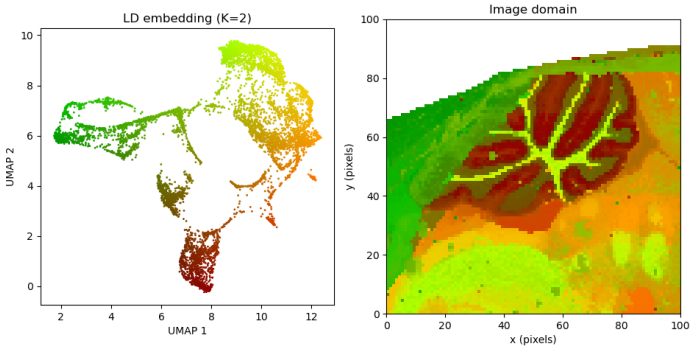
(a) Ion image of subsection (m/z value = 783.590)



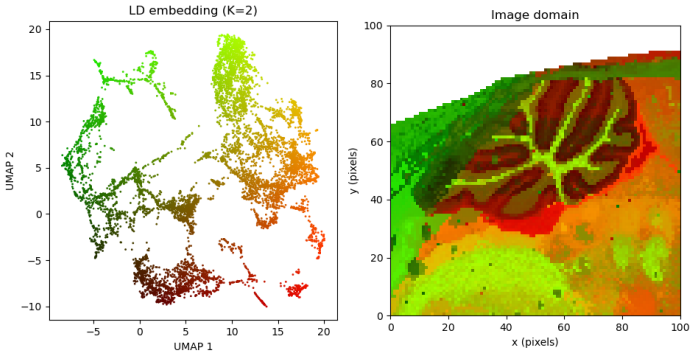
(b) PCA (explained variance ratio = 72.95%)



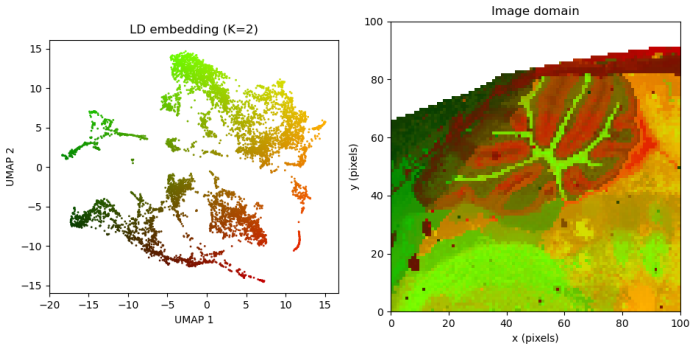
(c) t-SNE



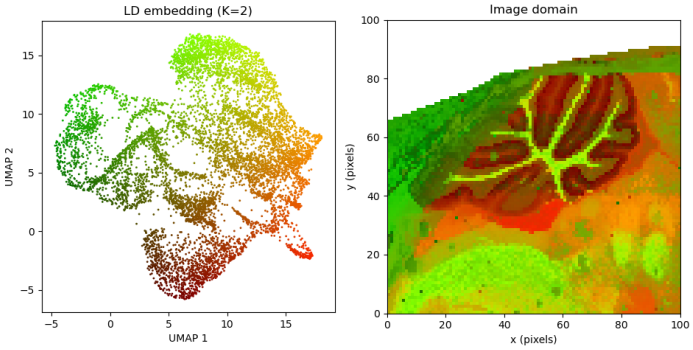
(d) UMAP



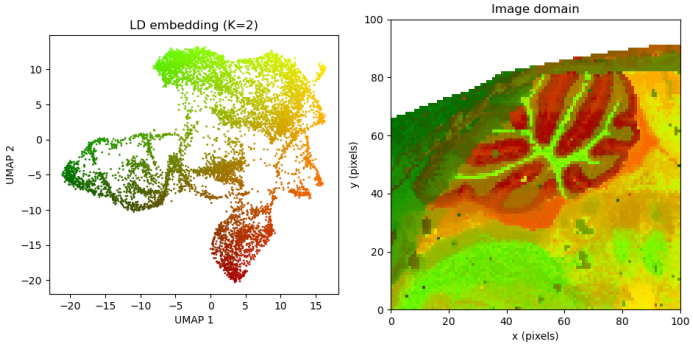
(e) DD-UMAP, 1-D optimization, `init`=spectral



(f) DD-UMAP, 1-D optimization, `init`=spatial



(g) DD-UMAP, Bayesian optimization, `init`=spectral



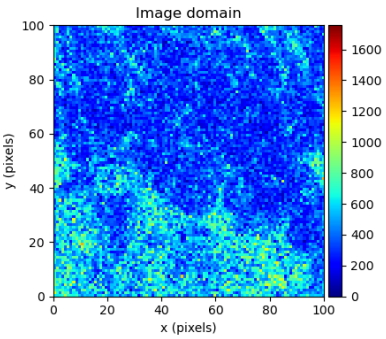
(h) DD-UMAP, Bayesian optimization, `init` = spatial

Figure 5-21: Comparison of multiple dimensionality reduction techniques on a subsection (brain) of the mouse pup data set. DD-UMAP is configured with both 1-D optimization and Bayesian optimization methods, and spectral and spatial initialization.`metric` = cosine, `n_epochs` = 100.

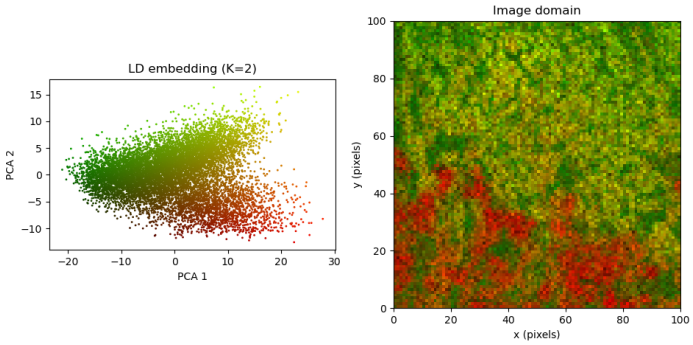
Kidney data set (subsection; outer layer)

A subsection in the kidney data set from the outside layer (cortex) is investigated. In Figure 5-22, an ion image of this subsection is visualized (with m/z value = 524.373). The PCA decomposition has more trouble in extracting the information compared to the subsection of the mouse brain, showing an explained variance ratio of 27.69% and a scattered LD embedding with two vaguely visible clusters. t-SNE shows unexpected behaviour by means of a few outliers, causing the colormap to only contain red-black values. Finer detail is lost. UMAP produces most details so far, visualizing 4 clusters defining the structure in the image domain. Similarly to the PCA decomposition, two clear regions are visualized (black under and green above). Besides these 2 regions, finer structure in the form of red-yellow make up the remainder of the image domain.

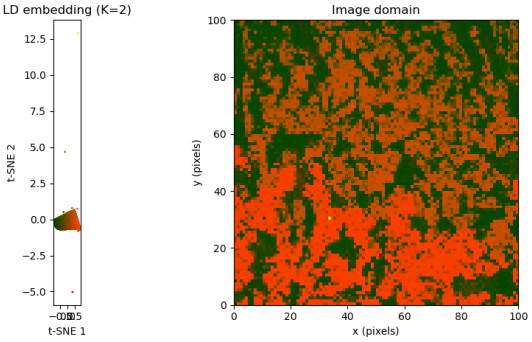
DD-UMAP has a reduced parameter range for `n_neighbors` ($= [2, 30]$) and `n_neighbors_spatial` ($= [4, 12, 28]$) since in all previous experiments, not better values for $CE_{combined}$ are obtained using high values. Moreover, the runtime decreases drastically for low values of these parameters, which becomes a crucial part for data sets of this size. Similar structures in the LD embeddings produced by DD-UMAP can be seen compared to regular-UMAP. The larger cluster size in Figure 5-22f causes a more subtle difference between the yellow-orange cluster, which has the red cluster as background in the image domain. Regular-UMAP in Figure 5-22d has a slightly different colormap, with the green cluster as background for the yellow and red cluster. This embedding indicates that the red and yellow cluster are more different from each other. The difference between the importance of dissimilarity can be brought down to the difference in initialization. Here, the spatial initialization places pixels closer to each other based upon spatial properties. Spatial initialization can be useful for pixels which may be similar, are close by each other, but the similarity is more difficult to extract from only spectral information.



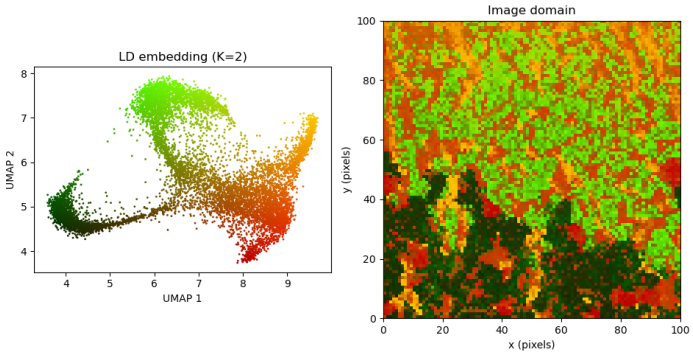
(a) Ion image of subsection (m/z value = 524.373)



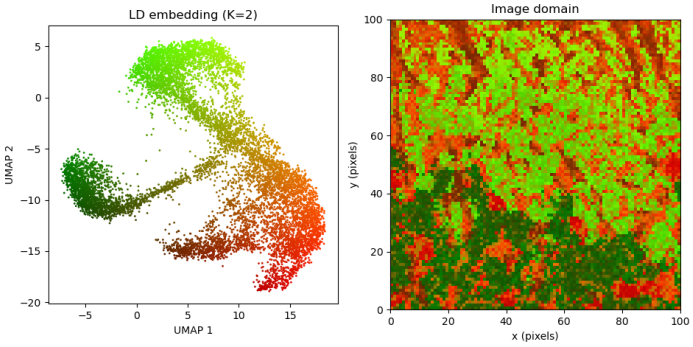
(b) PCA (explained variance ratio = 27.69%)



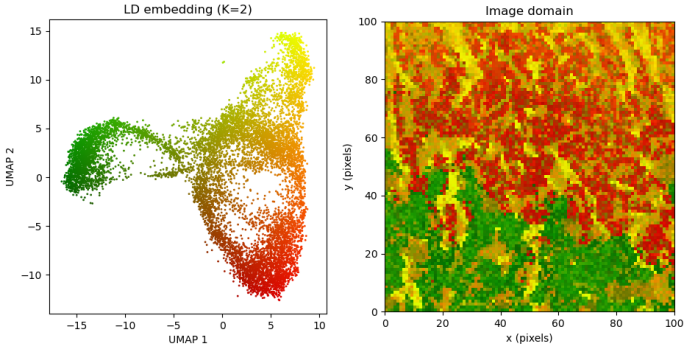
(c) t-SNE



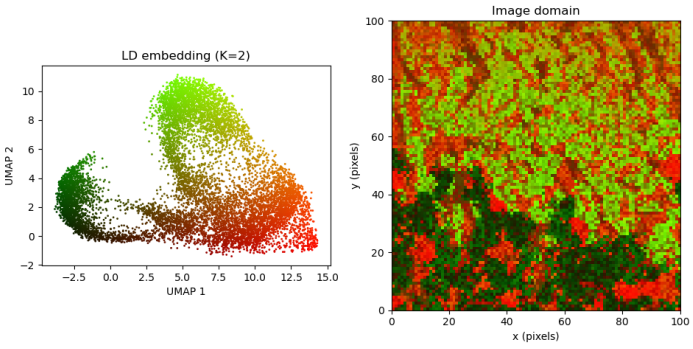
(d) UMAP



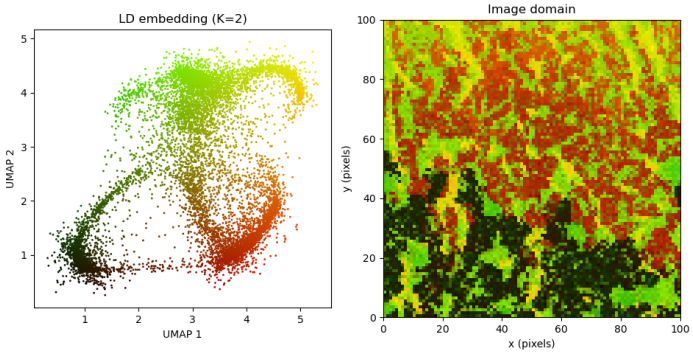
(e) DD-UMAP, 1-D optimization, `init`=spectral



(f) DD-UMAP, 1-D optimization, `init`=spatial



(g) DD-UMAP, Bayesian optimization, `init`=spectral

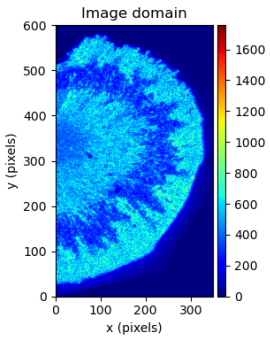


(h) DD-UMAP, Bayesian optimization, `init` = spatial

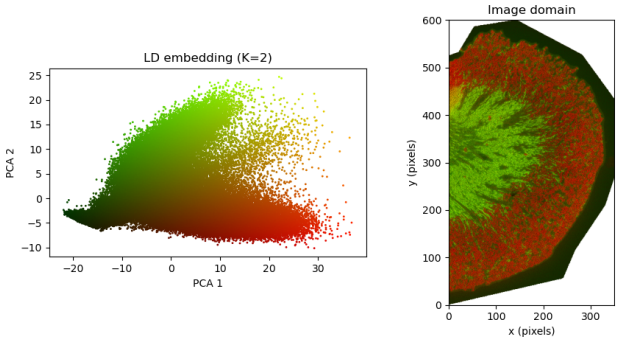
Figure 5-22: Comparison of multiple dimensionality reduction techniques on a subsection (outside layer) of the kidney data set. DD-UMAP is configured with both 1-D optimization and Bayesian optimization methods, and spectral and spatial initialization.`metric` = cosine, `n_epochs` = 100.

Kidney data set (subsection; right half)

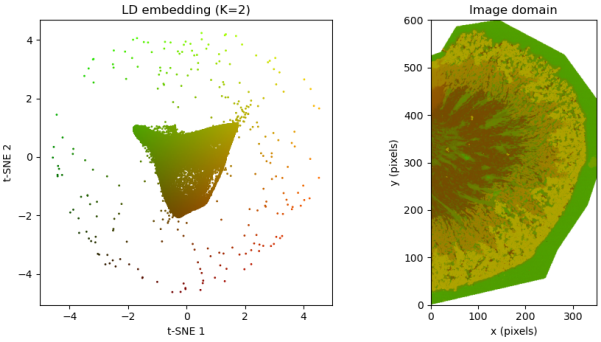
Finally, the complete right half of the kidney data set containing all large ROIs (renal inner medulla, outer medulla, and cortex) is explored. An ion image at m/z 524.373 in Figure 5-23a shows the indication of these layers already, but the differentiation becomes better visible in the PCA decomposition in Figure 5-23a with only 43.11% of explained variance. t-SNE has more difficulties with the separation, and the colormap is distorted due to the large amount of outliers. Although, now t-SNE finds an extra layer within the cortex. This layer is even more distinguishable in the image domain corresponding to regular-UMAP. More structure in the direction perpendicular to the radial curvature of the kidney is clearly present. The DD-UMAP using 1-D optimization and the spectral initialization finds a very similar embedding compared to regular-UMAP. Using spatial initialization in Figure 5-23f, similar effects seen in the smaller subsection of the kidney-cortex can be observed; the black accentuation of the cortex in Figure 5-23d and Figure 5-23e is less clearly visible due to the initial positioning of the pixels belonging to the cortex at close by positions in the LD embedding. The cluster corresponding to this accentuation is closer to other data points (when examining the LD embedding in Figure 5-23f). Both 1-D and Bayesian optimization methods produce comparable LD embeddings for similar initialization settings. Overall, the clusters produced by DD-UMAP are less separated compared to regular-UMAP, but are not substantially different from each other. This could indicate that the default settings for UMAP are sufficiently close to the optimum parameter configuration for this specific subsection of the data set.



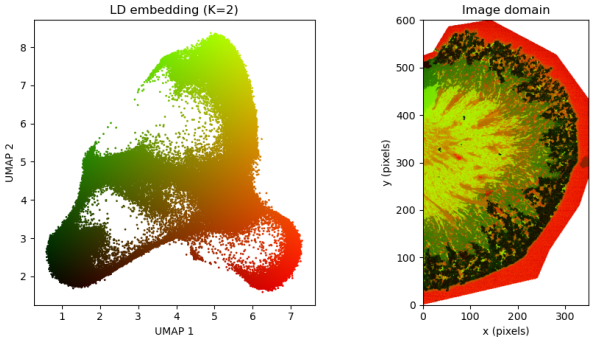
(a) Ion image of subsection (m/z value = 524.373)



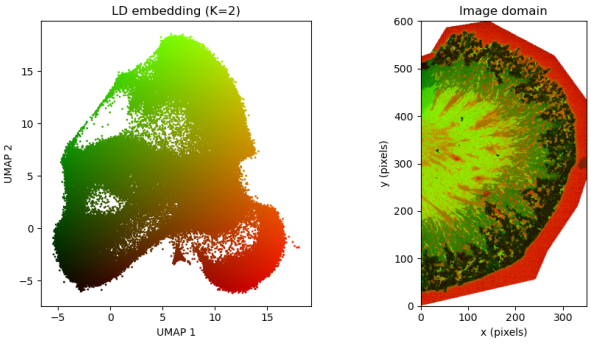
(b) PCA (explained variance ratio = 43.11%)



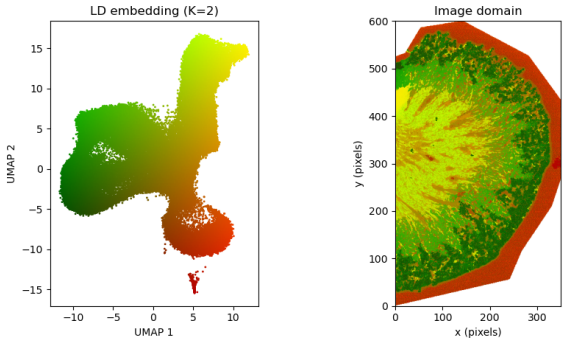
(c) t-SNE



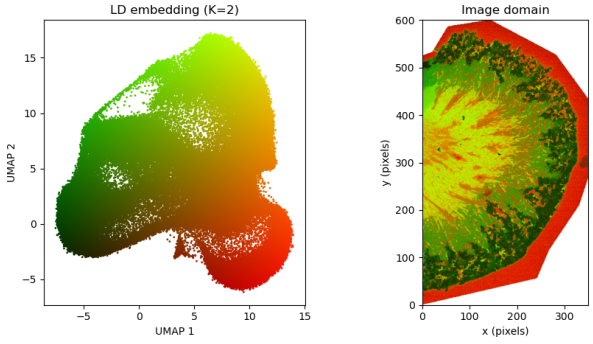
(d) UMAP



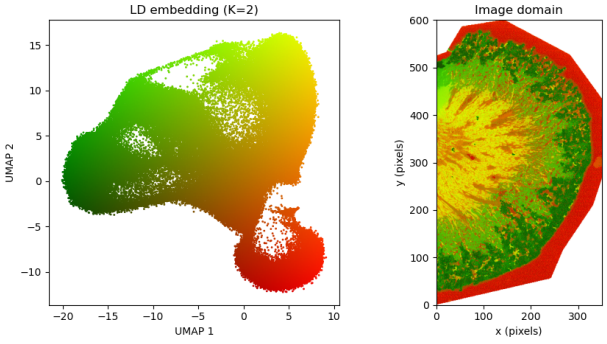
(e) DD-UMAP, 1-D optimization, `init`=spectral



(f) DD-UMAP, 1-D optimization, `init`=spatial



(g) DD-UMAP, Bayesian optimization, `init`=spectral



(h) DD-UMAP, Bayesian optimization, `init` = spatial

Figure 5-23: Comparison of multiple dimensionality reduction techniques on a subsection (right half) of the kidney data set. DD-UMAP is configured with both 1-D optimization and Bayesian optimization methods, and spectral and spatial initialization.`metric` = cosine, `n_epochs` = 100.

Chapter 6

Conclusions

6-1 Main conclusions

This thesis aims to take first steps in exploratory research on unsupervised dimensionality reduction techniques, and to provide evaluation tools for spatially structured data to take not only spectral but also spatial information into account. Two extensions of a NLDR technique titled UMAP [11] are proposed: UMAPLUS and DD-UMAP. UMAPLUS is able to take spatial location into account while optimizing its LD embedding, interpolating between spectral and spatial information through a variable parameter $\lambda_{spatial}$. Secondly, DD-UMAP optimizes the parameters of UMAP using either a 1-D optimization framework using Golden section search to find its optimum (one parameter at a time), or a multivariate Bayesian optimization approach capable of tuning multiple parameters at once. This chapter will go over the main conclusions extracted from the experiments conducted in this thesis.

Evaluation function $\mathbf{CE}_{combined}$ Data used in this thesis is unlabelled, providing a challenge to detect any structure. In order to evaluate the produced LD embedding, an evaluation function is created to take both spectral and spatial information into account. The pairwise distance matrices in high-dimensional space and LD embedding are used to compared the loss of information. Unfortunately, these matrices scale exponentially with the number of data points. Using the full pairwise distance matrices is unworkable due to the sheer amount of memory needed on the workstation. Therefore, subsampling is used to employ a subset of the data for evaluation, which has proven to be sufficient by a confidence score ϕ. Through a newly defined confidence score, the percentage of data used for subsampling is justified and used for every other subsequent experiment.

Using only $\text{CE}_{spectral}$ as evaluation scoring metric is possible, but unable to deal with the nonlinear mixing artificially added to the synthetic data set, rendering all multiplied spectra similar. On the other hand, using only $\text{CE}_{spatial}$ is completely undesirable, since the optimum LD embedding would be a 2-D plane with the shape of the the sampled tissue surface as basis, completely ignoring the chemical content of the pixels. Options such as

$CE_{combined,euclidean}$ and $CE_{combined,manhattan}$ have shown to favor spatial information more than desired. $CE_{combined}$ has been constructed, incorporating $\lambda_{spatial}$ to add more weights to the importance of spectral compared to spatial information.

Distance metric The Cosine distance metric has proven to be the most robust distance metric for the construction of the kNN-graph. Though a semi-grid search, it is shown that parts of the scoring metric $CE_{combined}$ ($CE_{spectral}$ and $CE_{spatial}$) produce the expected values when simultaneously inspecting the LD embedding and image domain as well.

Initialization Previous studies have argued the importance of the initialization phase of UMAP compared to t-SNE [96]. The initialization phase has therefore been reviewed, and besides the spectral and random initialization available within UMAP, the spatial initialization is proposed. This type of initialization showed similar results compared to the spectral initialization. On the synthetic data set without additive noise, the gradient is undetectable using the random initialization, better distinguishable using the spectral initialization, although the spatial initialization is able to recover the gradient the best. The added benefit is the alignment between the spectral and spatial domain; all elements found in the image domain can be directly mapped to the LD embedding. This becomes more clear on the synthetic data set with additive Gaussian noise ($\sigma = 10$), which clearly shows the expected positions of the nonlinear mixing in the LD embedding; between the circles where they were placed (overlapping pixels). The only downside would be that the spatial initialization is only possible for a 2-D LD embedding due to the nature of most IMS data sets (acquired from a 2-D tissue surface).

UMAPLUS and $\lambda_{spatial}$ UMAPLUS introduced $\lambda_{spatial}$ and the kNN-graph $\mathcal{G}_{spatial}$ constructed in the spatial domain, keeping track on the pixel locations with a predefined number of neighbors; `n_neighbors_spatial`. The effects of UMAPLUS on the LD embedding have been evaluated in experiment 3. Increasing `n_neighbors_spatial` effectively increases the number of forces used in the optimization of the LD embedding by UMAP of $\mathcal{G}_{spatial}$, thus causing a better connection between neighboring pixels in the image domain. The importance between optimizing the LD embedding using spectral and spatial information can be achieved through interpolation of $\lambda_{spatial}$; higher values cause larger forces using spatial information. This thesis aimed to find a balance between both sources of information. Using $CE_{combined}$, the desired balance can be achieved and it has resulted in a representation of the LD embedding smoothing out the nonlinear mixing in the overlapping regions of the synthetic data set.

DD-UMAP Lastly, a maximum number of 7 parameters have been optimized (5 from the original UMAP algorithm and 2 from UMAPLUS) using either a 1-D optimization method using Golden section search or a multivariate Bayesian optimization approach. Divided into 3 groups (small, medium and large framework), the optimization has been evaluated. The small framework only optimizes one parameter, which is partly done to showcase the working of both optimization methods. Both methods easily find the minimum while optimizing $\lambda_{spatial}$.

The medium framework imposed more difficulties for both frameworks. Since the 1-D optimization method only optimizes one parameter at a time, a large part of the parameter search

space remained undiscovered. The assumption was that the order of the parameters has a substantial effect on the found optimum parameter configuration. However, for the parameters in the medium framework (evaluated in experiment 4), only one order was substantially different from the rest (`spatial_lambda` → `n_neighbors` → `n_neighbors_spatial`). This resulted in almost a doubled value for $CE_{combined}$ compared to all other parameter sequences. This makes it difficult to rely on a single run produced by the 1-D optimization method, as it lacks robustness. Another disadvantage of the Golden section search is that it could potentially get stuck in a local minimum. This is shown in the semi-grid search of a subsection containing the brain of the mouse pup data set, varying `n_neighbors_spatial` (see Figure A-14). However, the speed of the 1-D optimization method compared to the Bayesian approach is a considerable reason to choose for the Golden section search.

The Bayesian optimization method also deviates slightly with multiple short runs, but remains more robust (compared to 1-D optimization) and can be relied on using only a single optimization cycle. Larger deviation between the optimum parameter configuration is observed in the large framework, which could indicate that the optimal LD embedding is not found yet and that more iterations are necessary. The vast parameter space results in lots of exploration by the Bayesian optimization method (e.g. sampling from areas of high uncertainties), which costs valuable iterations.

During the case study where real-world IMS data sets are investigated, DD-UMAP does extract more structure from the data compared to UMAP using default settings. In our experiments, $\lambda_{spatial}$ and `n_neighbors_spatial` corresponding to UMAPLUS were configured to a minimal value by DD-UMAP. The newly found structure mostly followed from a low value for `n_neighbors`, and high values for `min_dist` and `spread`. The low value for `n_neighbors` indicates that DD-UMAP favors local structure over global structure, as this results in a lower values for $CE_{combined}$.

6-2 Future work

This thesis proposed tools for utilizing spatial information in IMS, or any spatially structured data, and constructed a framework for optimizing multiple parameters in a sequential or multivariate unsupervised manner for dimensionality reduction by UMAP. Further extensions could potentially include:

- Generate a common evaluation function capable of numerically comparing LD embeddings produced by other dimensionality reduction techniques like PCA or t-SNE. The current approach utilized parameters computed by UMAP as building blocks of the evaluation function.
- Apply spatial initialization on 3-D IMS data sets, and subsequently UMAPLUS.
- An adjustable constraint on the parameter range during optimization could potentially expand the parameter search space, since `spread` must be bigger or equal to `min_dist`, therefore limiting further investigation.
- Interpolation between different types of initialization (e.g. spectral-spatial) allows a more local approach of forcing a subset of pixels to be placed near each other in the LD embedding, since they are spatially neighbors of each other.

- Investigate the necessary amount of iterations that the Bayesian optimization method needs for finding the optimal Low-Dimensional (LD) embedding, since the current implementation uses a fixed number of iterations scaling linear with the number of parameters.
- Apply multi-start or varying parameter bounds for the Golden section method in order to not get stuck in local minima during optimization.
- Apply weights to the runtime of the algorithm to limit the Bayesian optimization method from exploring regions of the parameter space that result in substantial runtimes.

Appendix A

Appendix

A-1 Parameters UMAP

Parameter name	Default value	Description
`n_neighbors`	15	Number of neighbours used per data point for constructing the kNN graph.
`n_components`	2	Dimension of the LD embedding.
`metric`	'euclidean'	Distance metric used in the high-dimensional space.
`metric_kwds`	None	Additional input argument for the ditance metric (e.g., the p-value for Minkowski distance)
`output_metric`	'euclidean'	Distance metric used in the low-dimensional embedding.
`output_metric_kwds`	None	Additional input argument for the distance metric (e.g., the p-value for Minkowski distance)
`n_epochs`	None	Number of training epochs for optimizing the low-dimensional embedding. In the UMAP source code, 500 epochs will be used for data sets with less than 10000 data points, and 200 epochs otherwise.
`learning_rate`	1.0	Initial learning rate for SGD.
`init`	'spectral'	Initialization of the LD embedding.
`min_dist`	0.1	Effective minimum distance between the data points in the low-dimensional embedding.

Table A-1: List of all parameters used by UMAP. Table continues: (1/3).

Parameter name	Default value	Description
spread	1.0	Effective scale of the data points in the low-dimensional embedding.
low_memory	True	Save memory during construction of kNN graph, by using more approximations in finding the nearest neighbors.
n_jobs	-1	The number of jobs to run in parallel. All cores are used for n_jobs $= -1$
set_op_mix_ratio	1.0	Symmetrization: ratio between (fuzzy) union and intersection. 1.0 result in fuzzy union, 0.0 result in pure fuzzy intersection.
local_connectivity	1.0	Number of nearest neighbors that should be assumed to be connected at a local level.
repulsion_strength	1.0	Weighting applied to the negative samples in optimizing the low-dimensional embedding.
negative_sample_rate	5	Number of negative samples to select for each positive sample during the optimization phase.
transform_queue_size	4.0	Controls how aggressively to search for nearest neighbors on new data, appended to the already trained model.
a	None	Models the distance probabilities in the LD embedding. This parameter followed from nonlinear least-square fitting by min_dist and spread, but also can be set manually.
b	None	Models the distance probabilities in the LD embedding. This parameter followed from nonlinear least-square fitting by min_dist and spread, but also can be set manually.
random_state	None	Seed that is used by the random number generator. For reproducible results, users should use one value throughout their research.
angular_rp_forest	False	Whether to use an angular random projection forest to initialise the approximate nearest neighbor search
target_n_neighbors	-1	The number of nearest neighbors to use to construct the target simplicial set. target_n_neighbors $= -1$ uses the n_neighbors value.
target_metric	'categorical'	Distance metric used on the target data during supervised dimensionality reduction.
target_metric_kwds	None	Additional input argument for the target distance metric (e.g., the p-value for Minkowski distance)
target_weight	0.5	Weighting factor between the existing data topology and (desired) target topology.
transform_seed	42	Random seed used for the stochastic aspects of the transform operation.
transform_mode	'embedding'	Returns either the embedding or kNN graph during transformation of the data.

Table A-2: List of all parameters used by UMAP. Table continues: (2/3).

Parameter name	Default value	Description
`force_approximation_algorithm`	False	Calculate subset of the forces applied by UMAP for data with less than 4096 data points.
`verbose`	False	Whether to print status data during the computation.
`unique`	False	Controls whether to remove duplicate data points from the data before computing LD embedding.
`densmap`	False	Specifies whether the additional density preserving term will be incorporated into the cost function. True will lead to Equation 3-9.
`dens_lambda`	2.0	Controls the density preservation term of densMAP.
`dens_frac`	0.3	Controls the fraction of epochs that are being used for densMAP. The first $1-$ `dens_frac` epochs will be used for regular UMAP.
`dens_var_shift`	0.1	A small constant added to the variance of local radii in the embedding when calculating the density correlation objective to prevent numerical instability from dividing by a small number.
`output_dens`	False	Returns local radii of the final LD embedding when set to True, computed by densMAP.
`disconnection_distance`	0.3	Disconnect any vertices of distance greater than or equal to `disconnection_distance` when approximating the manifold by construction of the kNN graph.

Table A-3: List of all parameters used by UMAP. Table continues: (3/3).

A-2 Semi grid search through parameters UMAP

This section will provide extensive visualizations of the LD embedding obtained by UMAP and the corresponding image domain, using a semi grid search varying one parameter for various distance metrics for the construction of the kNN-graph $\mathcal{G}_{spectral}$ while keeping all other parameters at their default configurations. Some parameters have been overridden: the `pixel_order` and `img_shape` have been updated based on the input data, `random_state` $= 42$ and `n_epochs` $= 100$. The LD embedding and image domain with default settings for UMAP should produce identical results. But, since $\mathcal{G}_{spectral}$ is not locally connected for the synthetic data set using the Euclidean distance metric, the spectral initialization may not work as expected. These results in an unrepeatable optimization of UMAP. The cosine distance is fully connected, hence resulting in reproducible results.

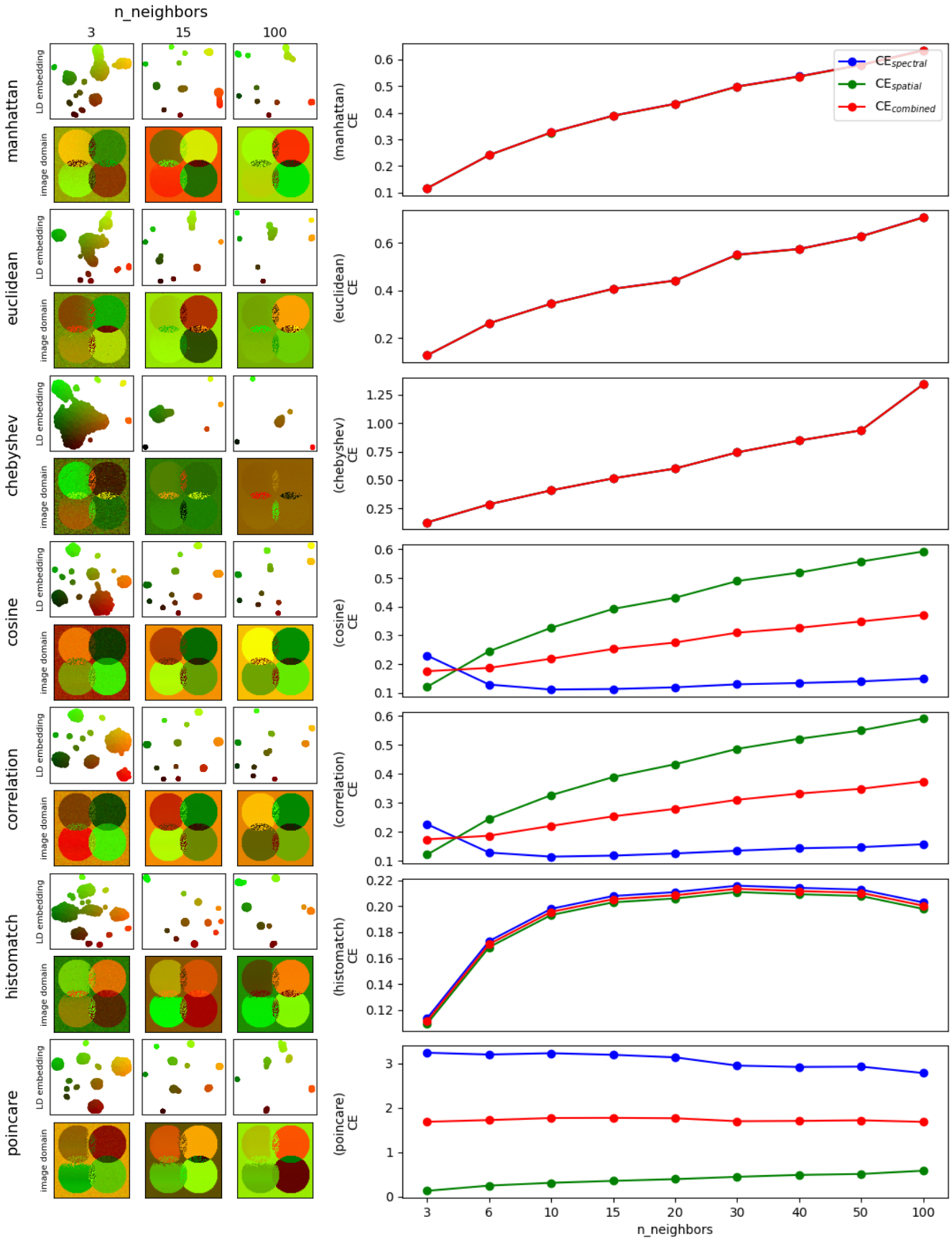


Figure A-1: Varying `n_neighbors`, **(a)** LD embedding and the corresponding image domain for various distance metrics, and **(b)** The evolution of $CE_{spectral}$, $CE_{spatial}$ and $CE_{combined}$ (lower values are better).

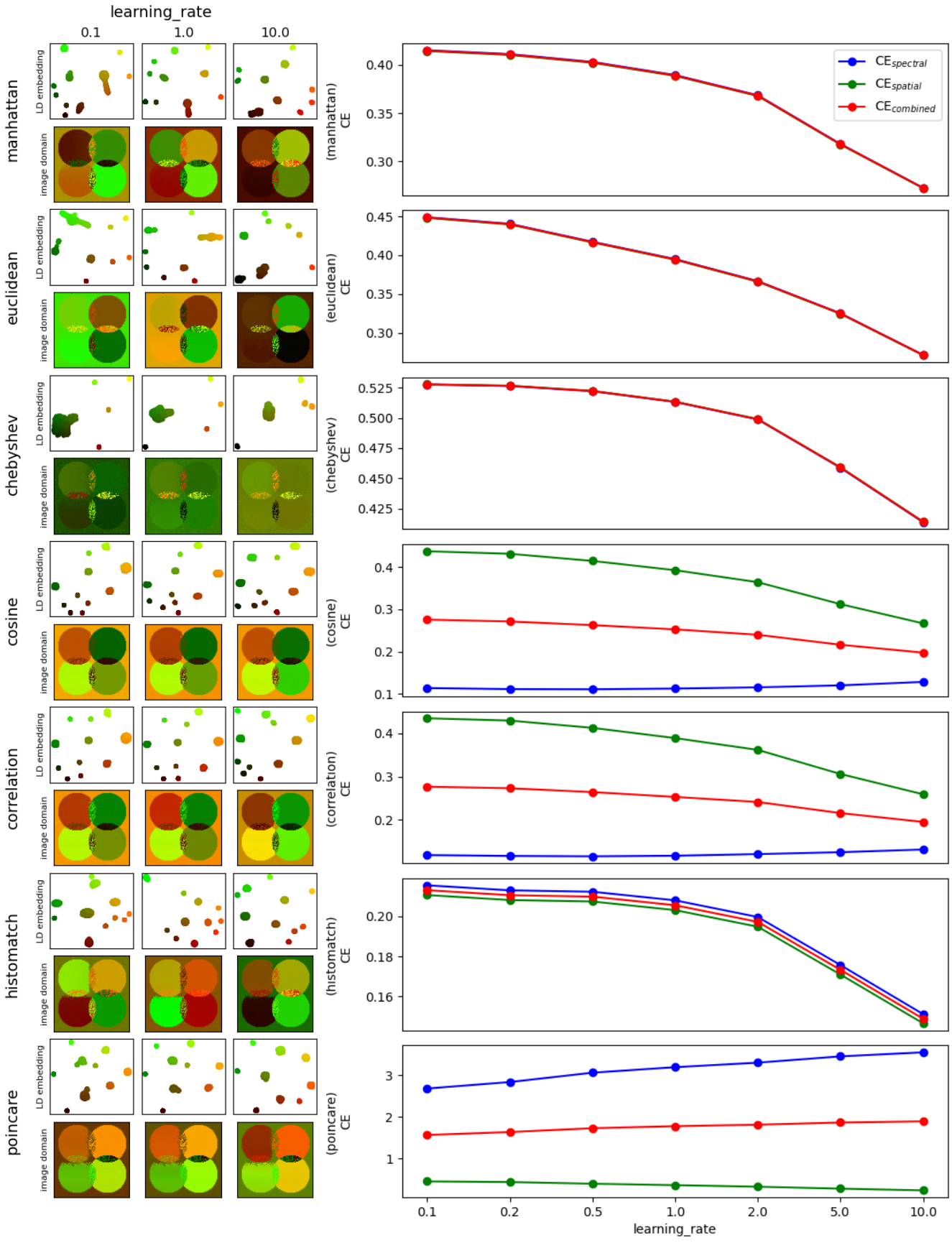
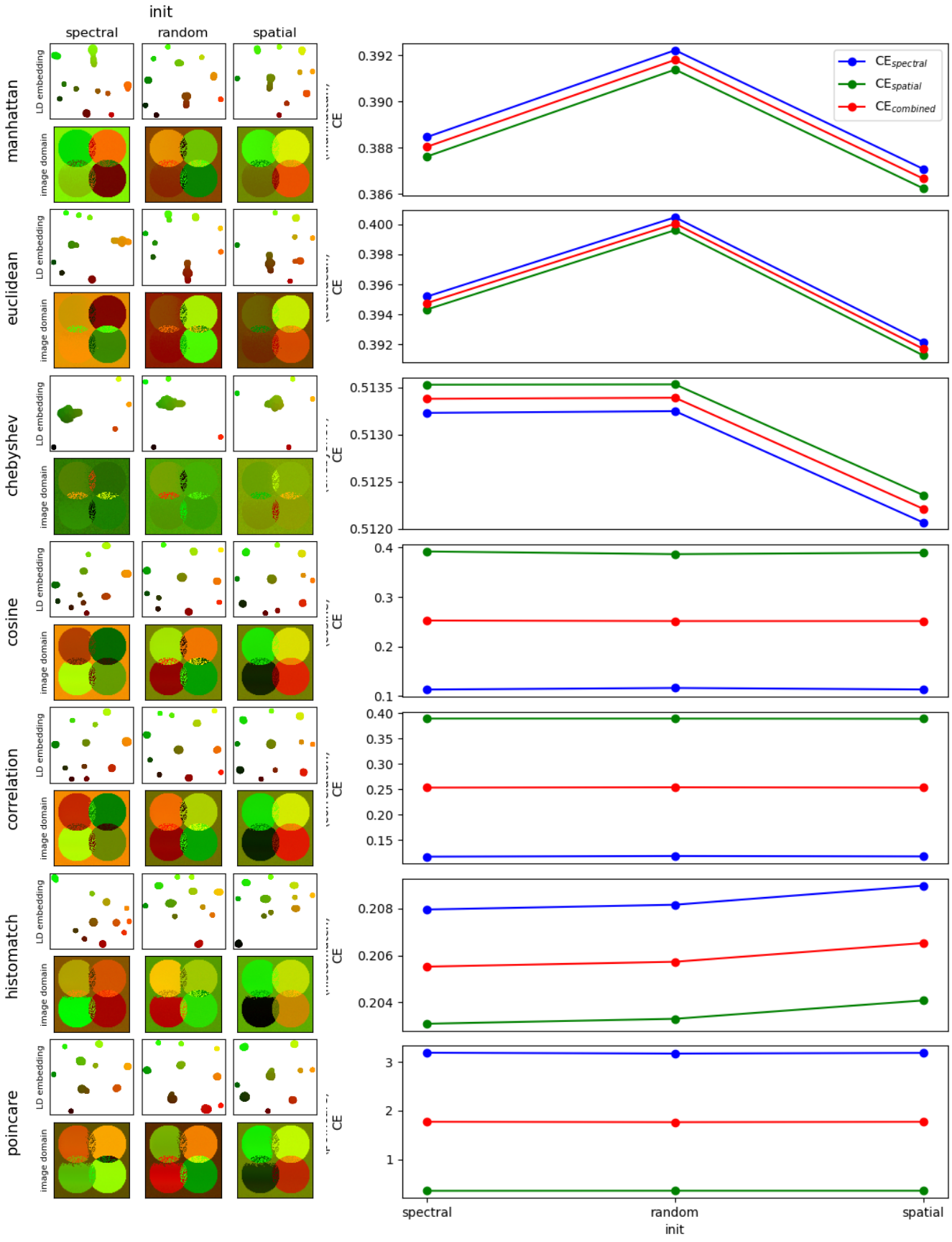


Figure A-2: Varying `learning_rate`, **(a)** LD embedding and the corresponding image domain for various distance metrics, and **(b)** The evolution of $CE_{spectral}$, $CE_{spatial}$ and $CE_{combined}$ (lower values are better).



(a) **(b)**

Figure A-3: Varying `init`, **(a)** LD embedding and the corresponding image domain for various distance metrics, and **(b)** The evolution of $CE_{spectral}$, $CE_{spatial}$ and $CE_{combined}$ (lower values are better).

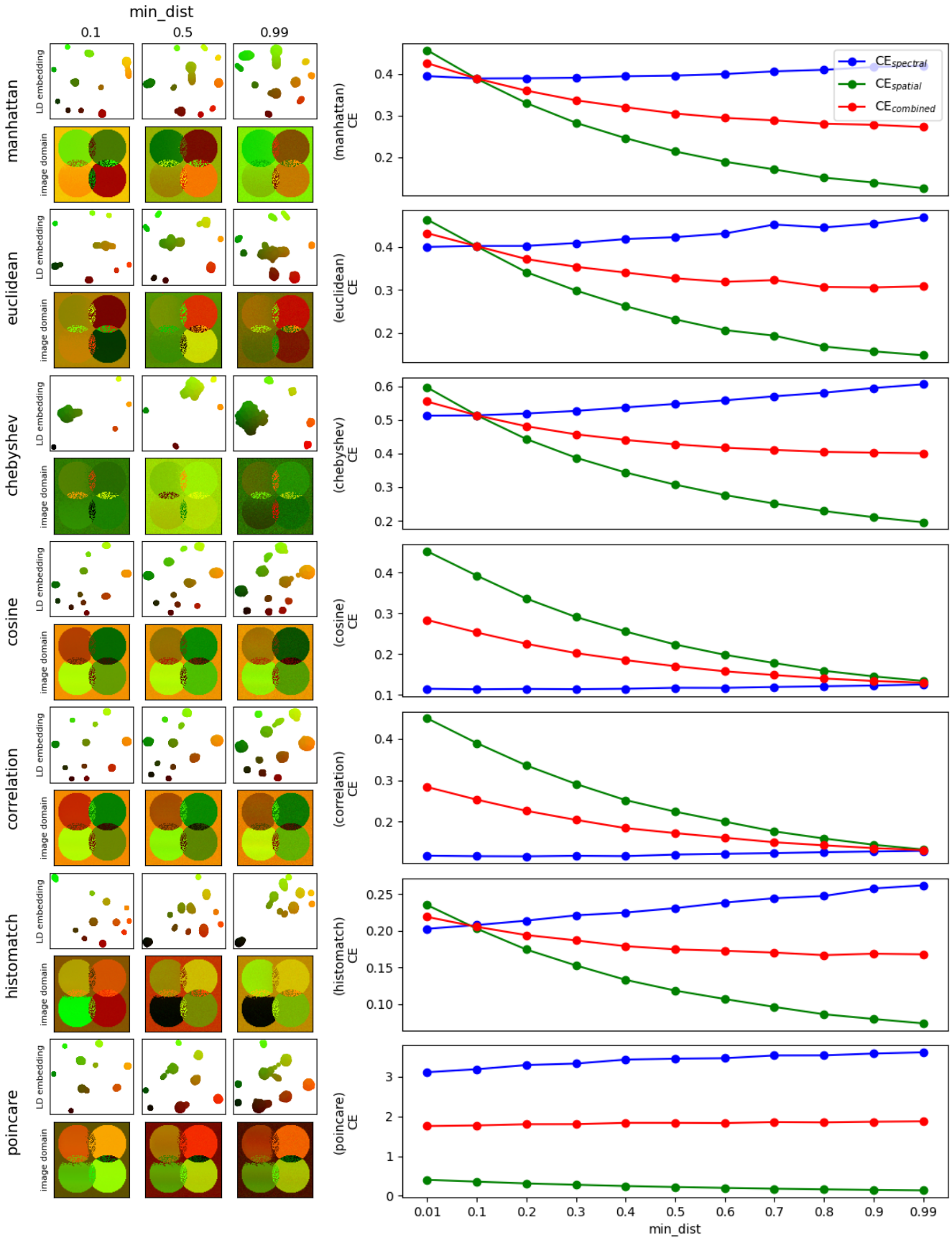


Figure A-4: Varying `min_dist`, **(a)** LD embedding and the corresponding image domain for various distance metrics, and **(b)** The evolution of $CE_{spectral}$, $CE_{spatial}$ and $CE_{combined}$ (lower values are better).

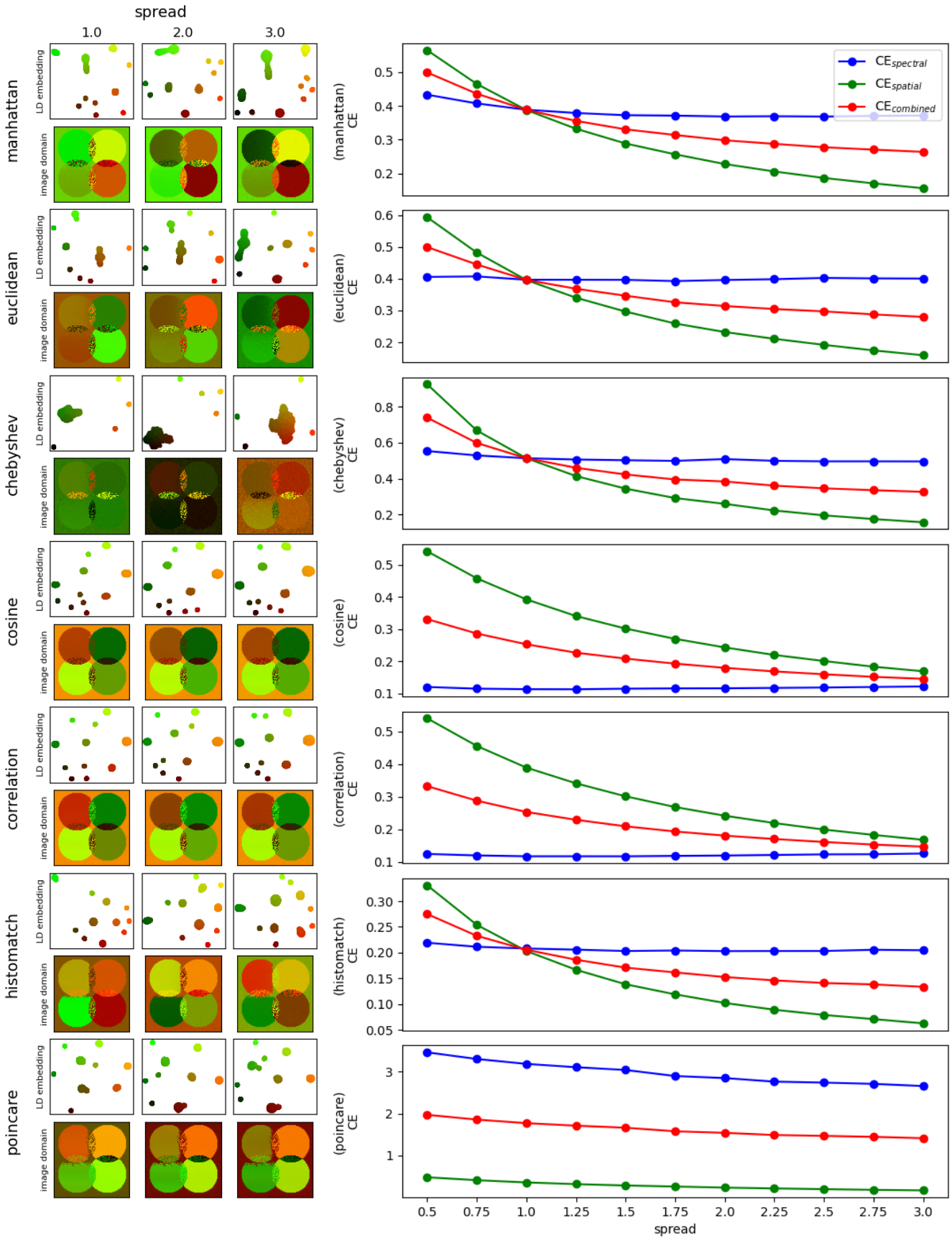


Figure A-5: Varying `spread`, **(a)** LD embedding and the corresponding image domain for various distance metrics, and **(b)** The evolution of $\text{CE}_{spectral}$, $\text{CE}_{spatial}$ and $\text{CE}_{combined}$ (lower values are better).

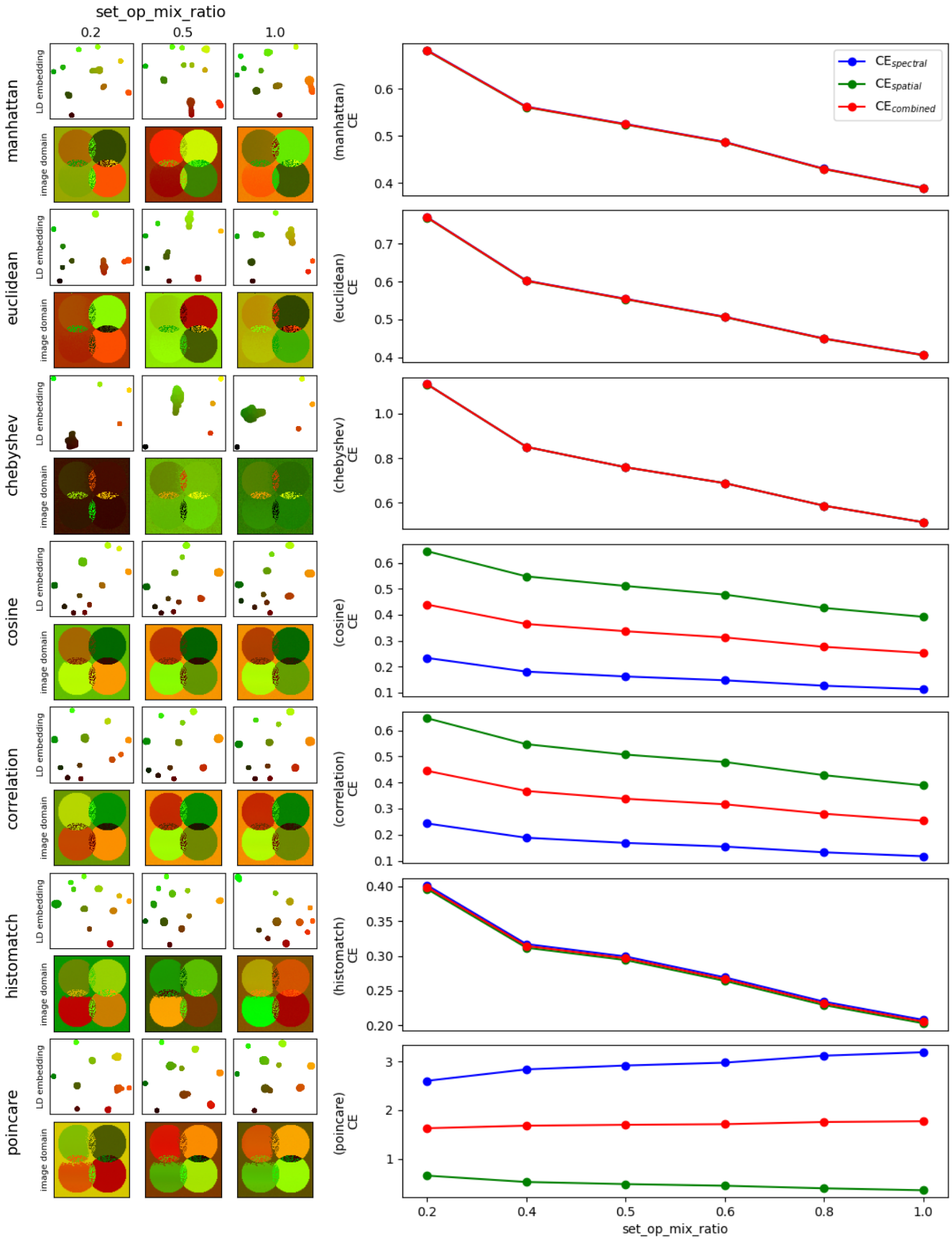


Figure A-6: Varying `set_op_mix_ratio`, **(a)** LD embedding and the corresponding image domain for various distance metrics, and **(b)** The evolution of $\text{CE}_{spectral}$, $\text{CE}_{spatial}$ and $\text{CE}_{combined}$ (lower values are better).

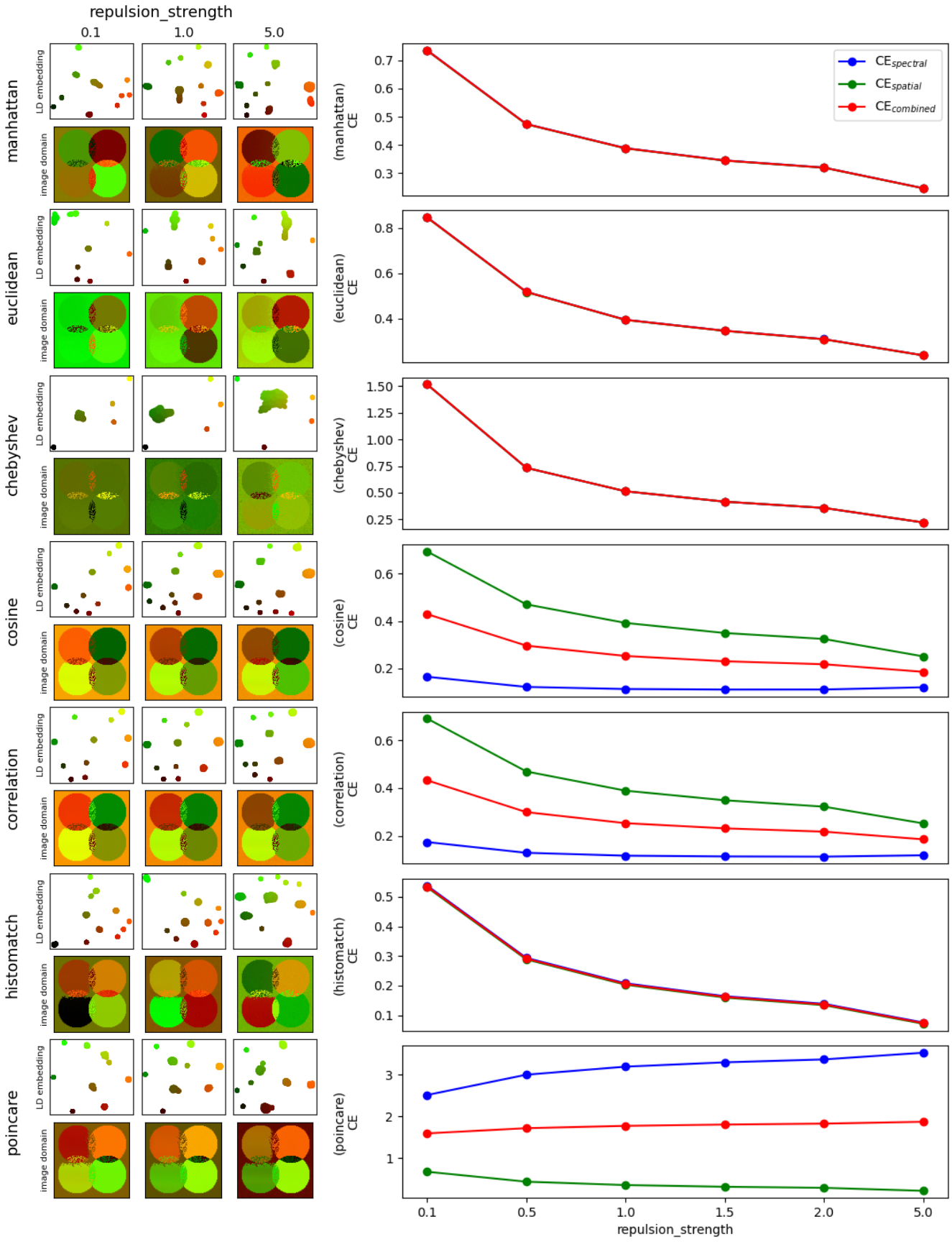


Figure A-7: Varying `repulsion_strength`, **(a)** LD embedding and the corresponding image domain for various distance metrics, and **(b)** The evolution of $CE_{spectral}$, $CE_{spatial}$ and $CE_{combined}$ (lower values are better).

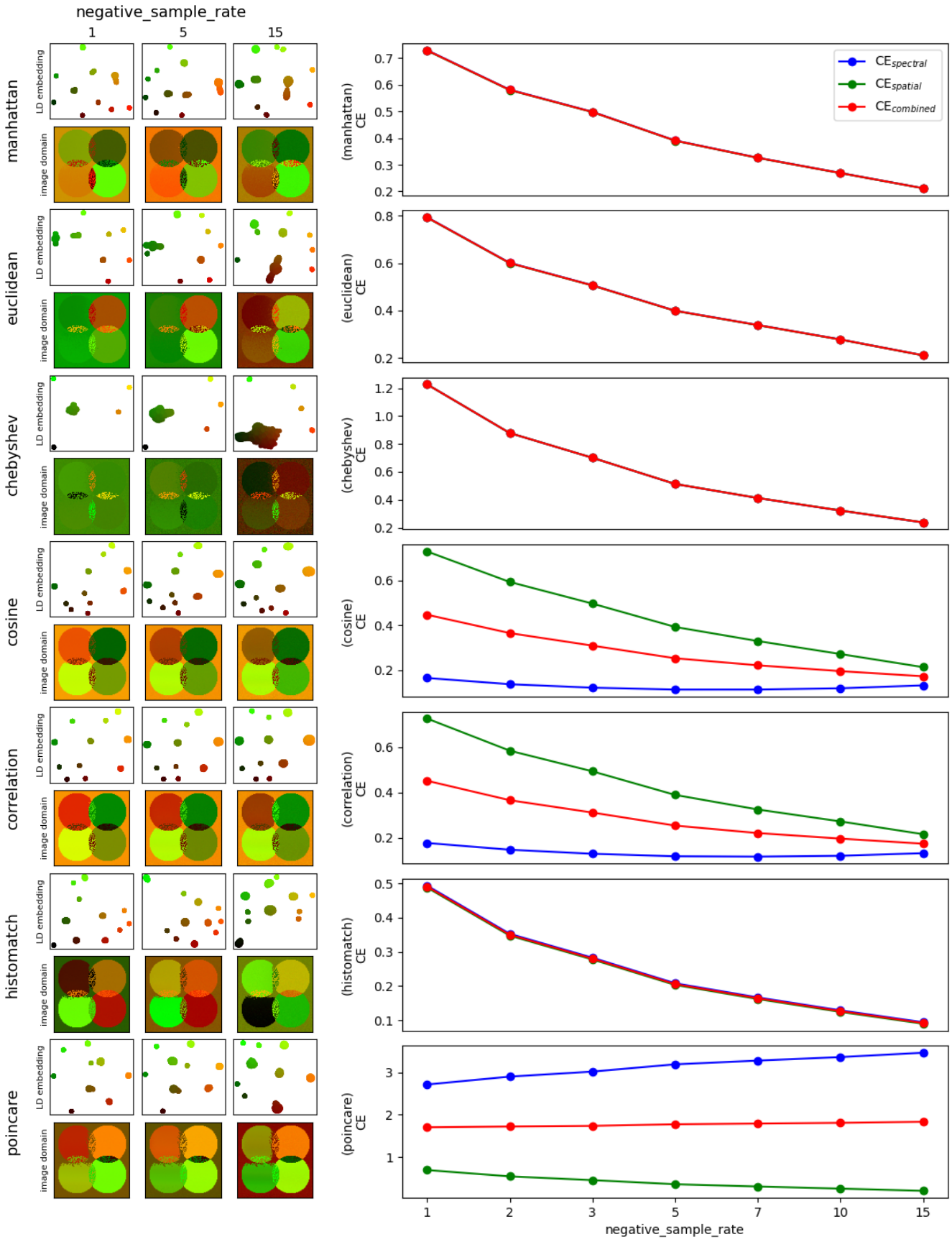


Figure A-8: Varying `negative_sample_rate`, **(a)** LD embedding and the corresponding image domain for various distance metrics, and **(b)** The evolution of $CE_{spectral}$, $CE_{spatial}$ and $CE_{combined}$ (lower values are better).

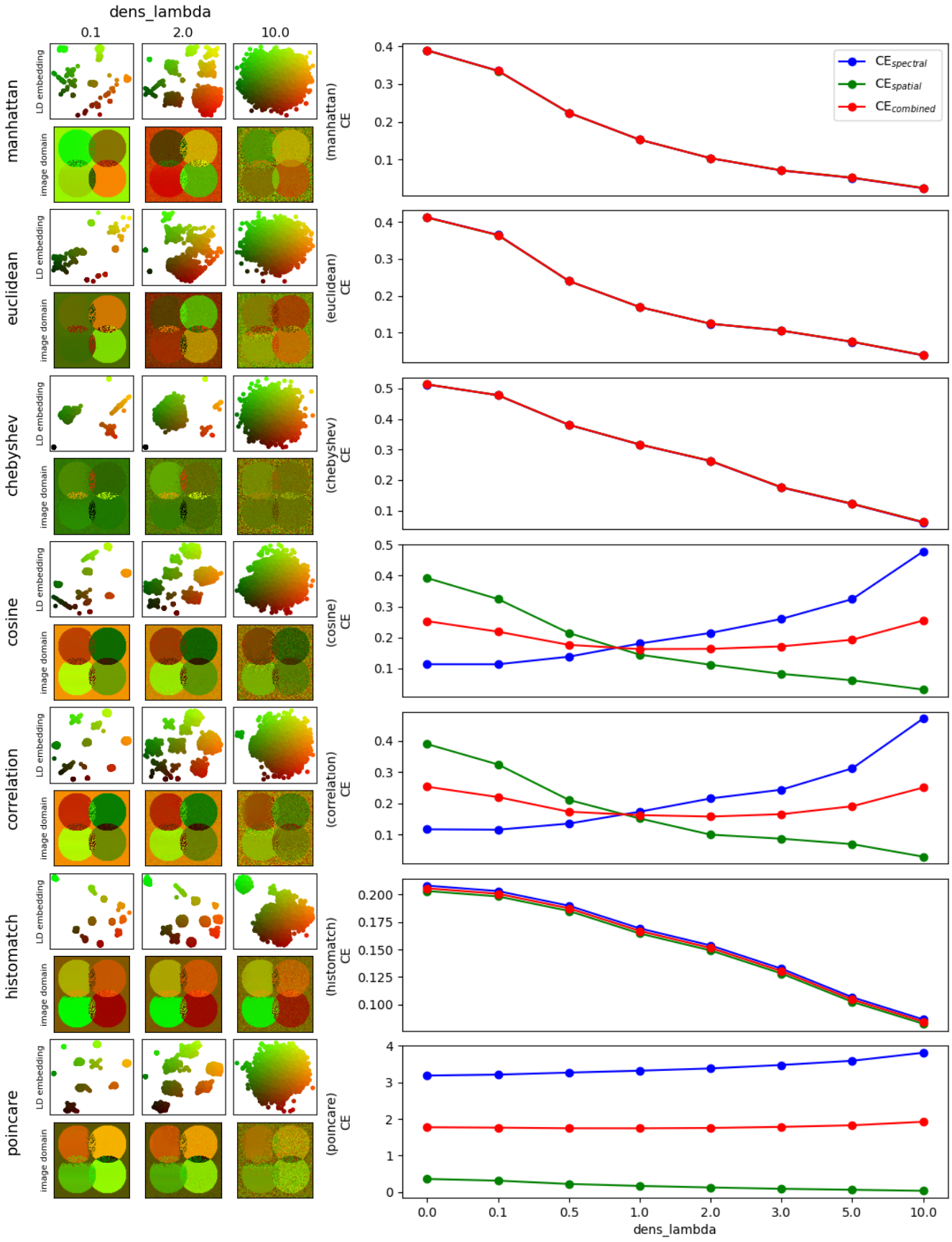


Figure A-9: Varying `dens_lambda`, **(a)** LD embedding and the corresponding image domain for various distance metrics, and **(b)** The evolution of $CE_{spectral}$, $CE_{spatial}$ and $CE_{combined}$ (lower values are better).

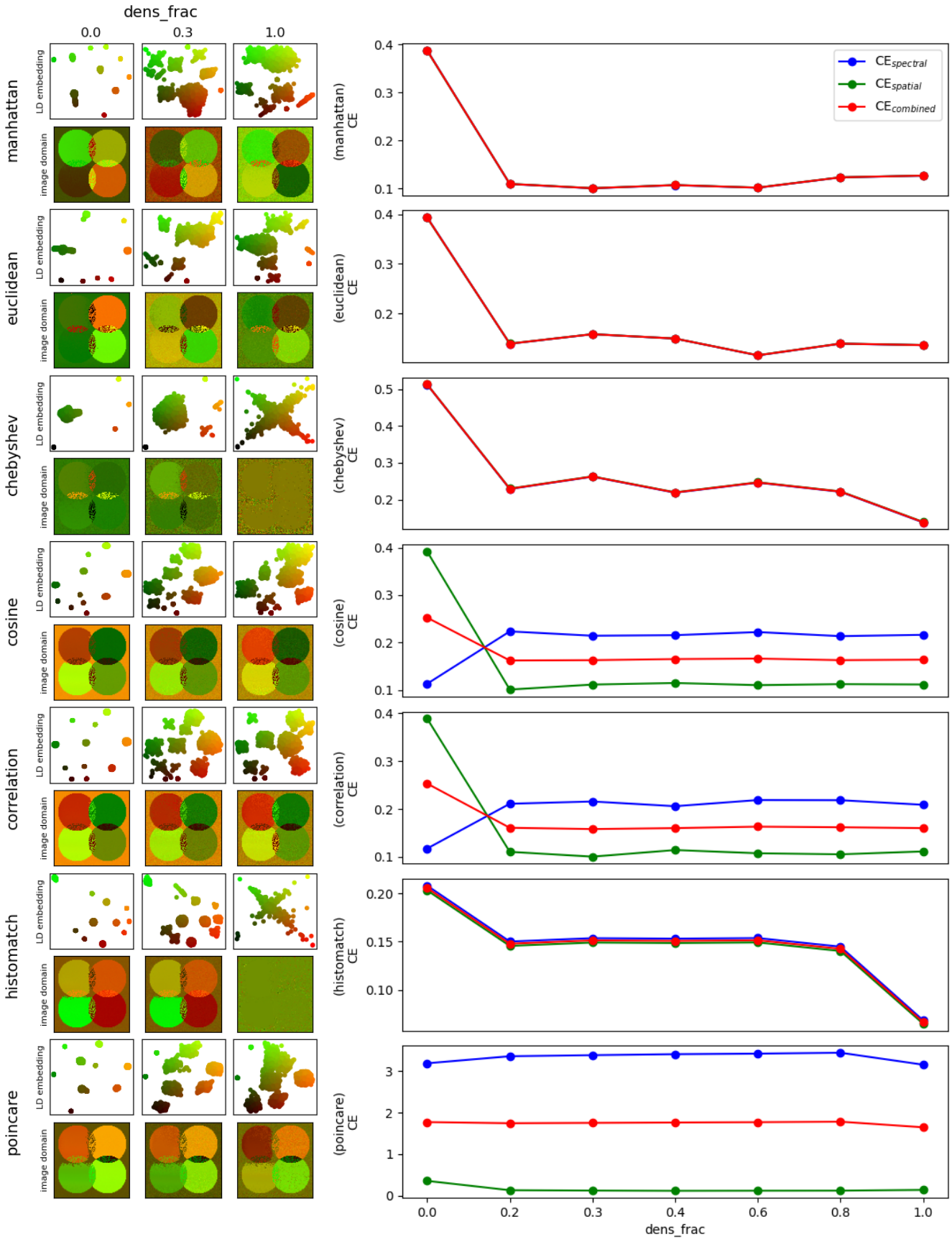


Figure A-10: Varying `dens_frac`, **(a)** LD embedding and the corresponding image domain for various distance metrics, and **(b)** The evolution of $CE_{spectral}$, $CE_{spatial}$ and $CE_{combined}$ (lower values are better).

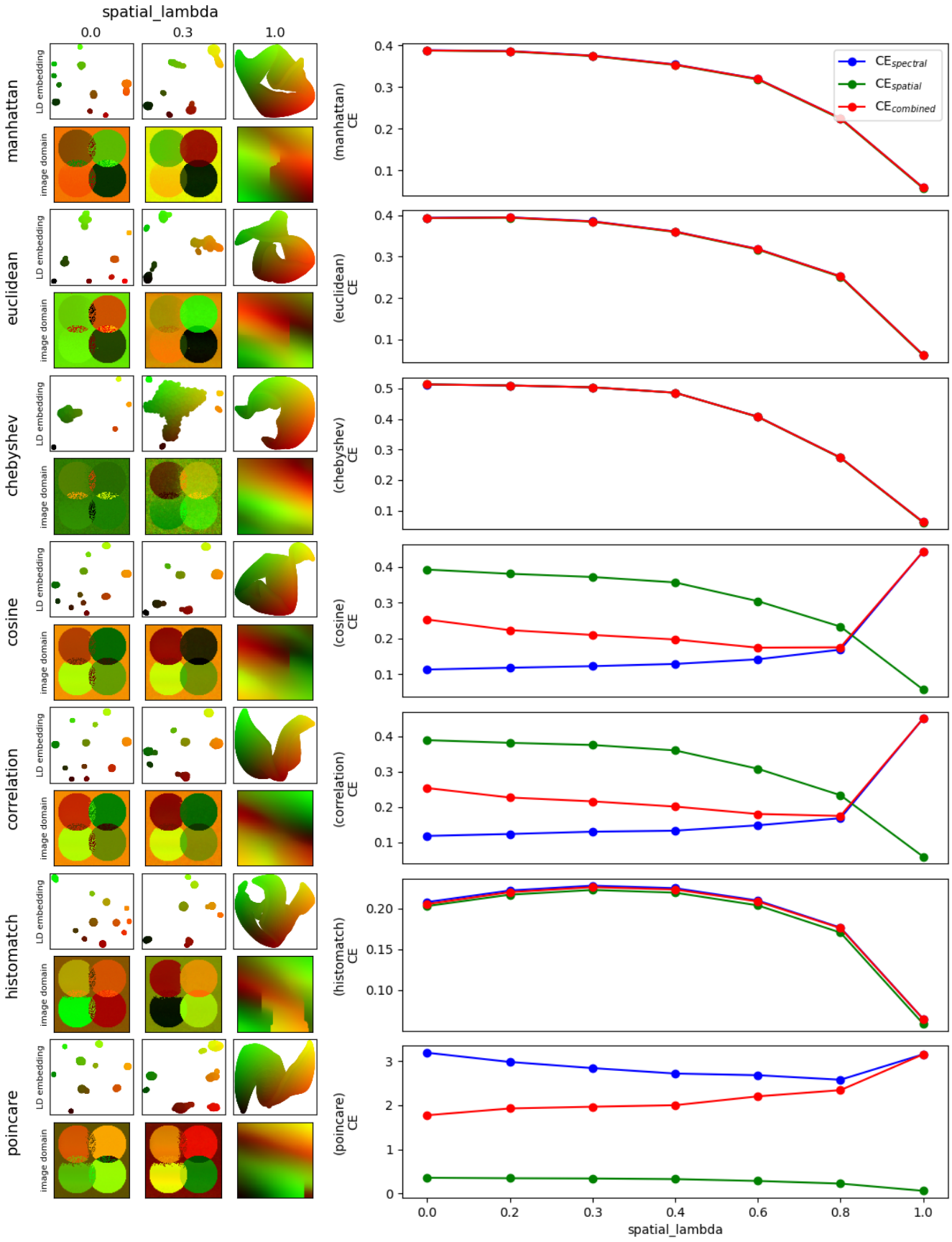


Figure A-11: Varying `spatial_lambda`, **(a)** LD embedding and the corresponding image domain for various distance metrics, and **(b)** The evolution of CE$_{spectral}$, CE$_{spatial}$ and CE$_{combined}$ (lower values are better).

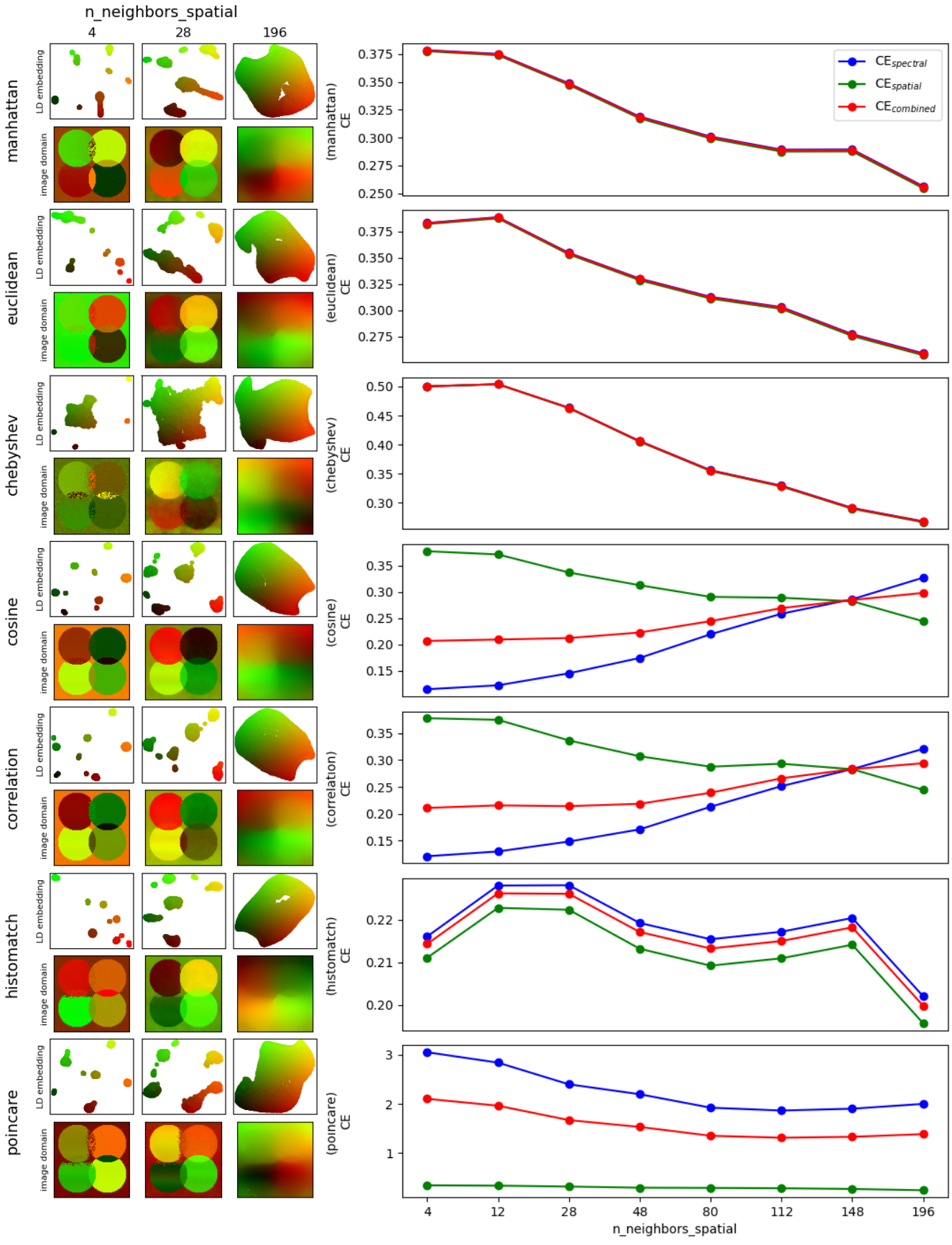


Figure A-12: Varying `n_neighbors_spatial`, **(a)** LD embedding and the corresponding image domain for various distance metrics, and **(b)** The evolution of $CE_{spectral}$, $CE_{spatial}$ and $CE_{combined}$ (lower values are better).

A-2-1 Fixed distance metric (cosine)

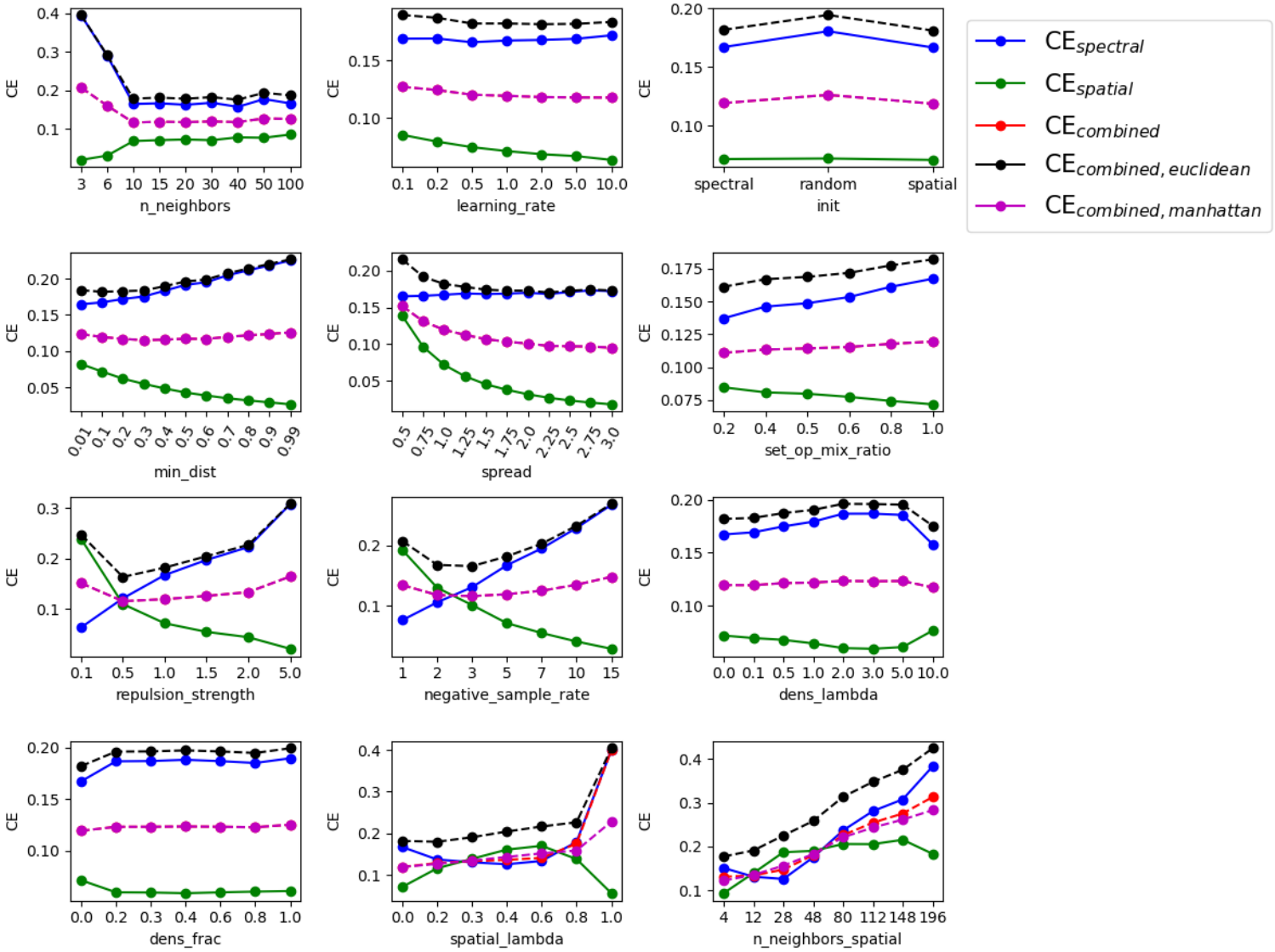


Figure A-13: Semi grid search on the synthetic data set without additive noise over the most important parameters of UMAP (lower values are better). Fixed variables: `metric` = *cosine*, `n_epochs` = 100 and `random_state` = 42.

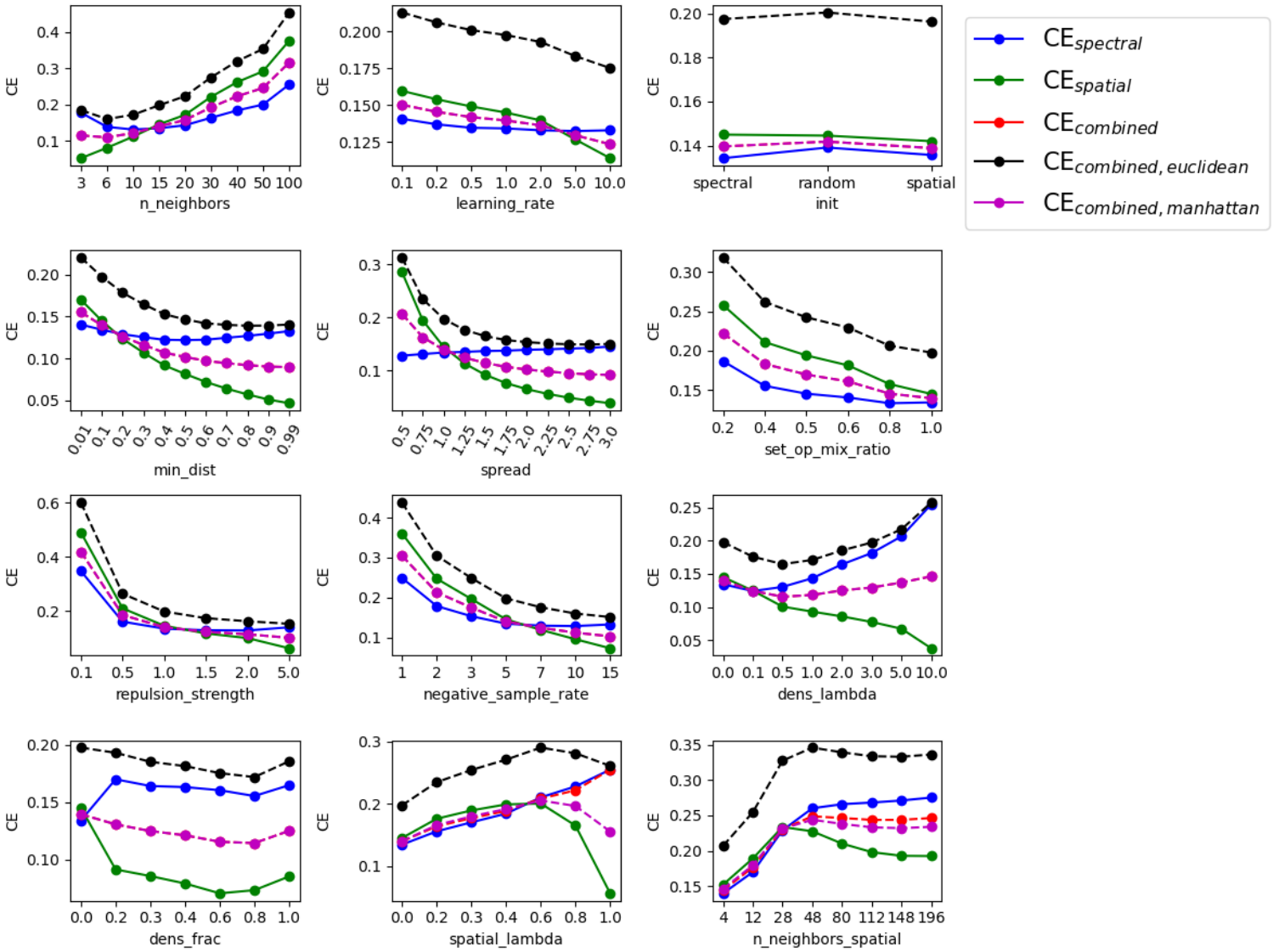


Figure A-14: Semi grid search on a subsection of the brain of the mouse pup data set over the most important parameters of UMAP (lower values are better). Fixed variables: `metric` = *cosine*, `n_epochs` = 100 and `random_state` = 42.

A-3 Results of optimization methods on different parameter frameworks

	Parameter order	Final CE	Runtime (min)	Final parameter configuration						
				spatial_lambda	n_neighbors	n_neighbors_spatial	learning_rate	min_dist	spread	set_op_mix_ratio
Small framework	Independent	0.169	4.16	0.711	-	-	-	-	-	-
Medium framework	Independent	0.209	12.95	0.711	4	12	-	-	-	-
	1 - 2 - 3	0.374	12.64	0.711	4	112	-	-	-	-
	1 - 3 - 2	0.172	12.36	0.711	4	12	-	-	-	-
	2 - 1 - 3	0.175	9.14	0.333	4	12	-	-	-	-
	2 - 3 - 1	0.175	9.47	0.333	4	12	-	-	-	-
	3 - 1 - 2	0.172	12.87	0.711	4	12	-	-	-	-
	3 - 2 - 1	0.175	10.76	0.333	4	12	-	-	-	-
Large framework	Independent	0.199	21.43	0.711	4	12	9.107	0.944	2.889	0.95]
	6 - 1 - 7 - 3 - 5 - 4 - 2	0.102	15.75	0.711	4	112	8.556	0.618	2.820	0.65
	6 - 1 - 5 - 7 - 3 - 2 - 4	0.094	29.75	0.012	5	28	7.663	0.854	2.820	0.78
	1 - 7 - 2 - 5 - 3 - 6 - 4	0.091	16.21	0.031	7	28	7.663	0.910	2.820	0.91
	6 - 7 - 5 - 4 - 1 - 2 - 3	0.090	17.43	0.031	8	12	8.556	0.910	2.820	0.91
	7 - 1 - 2 - 4 - 5 - 3 - 6	0.092	16.37	0.044	8	12	8.556	0.910	2.820	0.95
	2 - 3 - 1 - 5 - 7 - 4 - 6	0.093	15.82	0.065	8	12	6.219	0.910	2.820	0.95
	7 - 3 - 5 - 6 - 1 - 2 - 4	0.089	15.78	0.065	8	12	6.219	0.910	2.820	0.95
	4 - 1 - 5 - 3 - 7 - 2 - 6	0.093	15.50	0.065	8	12	6.219	0.910	2.820	0.95
	6 - 1 - 7 - 2 - 4 - 5 - 3	0.089	15.17	0.065	8	12	6.219	0.910	2.820	0.95
	7 - 1 - 4 - 3 - 2 - 5 - 6	0.093	15.10	0.065	8	12	6.219	0.910	2.820	0.95

Table A-4: 1-D optimization: overview of different order of optimizing 9 parameters of UMAP. After each optimization cycle of one parameter, the optimum value for that parameter is saved and used for subsequent optimization cycles. Except for the *independent* case, which starts with the default settings of the parameter at each optimization cycle. The displayed values are the best found configurations during the allowable amount of iterations. For the small and medium framework are only 1 and 6 parameter orders possible, respectively. The large framework can contain up to $7! = 5,040$ possibilities, but only 10 are chosen here (randomly). The number are abbreviations to the parameters: 1 : `spatial_lambda`, 2 : `n_neighbors`, 3 : `n_neighbors_spatial`, 4 : `learning_rate`, 5 : `min_dist`, 6 : `spread`, 7 : `set_op_mix_ratio`.

	random_state	Final CE	Runtime (min)	Final parameter configuration						
				spatial_lambda	n_neighbors	n_neighbors_spatial	learning_rate	min_dist	spread	set_op_mix_ratio
Small framework (8 iterations)	42	0.174	4.19	0.782	-	-	-	-	-	-
	43	0.174	4.15	0.616	-	-	-	-	-	-
	44	0.172	4.05	0.683	-	-	-	-	-	-
	45	0.170	3.89	0.722	-	-	-	-	-	-
	46	0.169	3.86	0.722	-	-	-	-	-	-
	47	0.181	3.86	0.825	-	-	-	-	-	-
	48	0.171	3.78	0.697	-	-	-	-	-	-
	49	0.170	3.79	0.742	-	-	-	-	-	-
	50	0.174	3.79	0.614	-	-	-	-	-	-
	51	0.178	3.76	0.550	-	-	-	-	-	-
Medium framework (24 iterations)	42	0.175	35.07	0.758	44	12	-	-	-	-
	43	0.181	75.56	0.502	12	12	-	-	-	-
	44	0.151	116.22	0.887	79	4	-	-	-	-
	45	0.152	163.38	0.923	20	4	-	-	-	-
	46	0.190	38.81	0.055	5	28	-	-	-	-
	47	0.151	78.16	0.970	59	4	-	-	-	-
	48	0.162	119.30	0.812	52	4	-	-	-	-
	49	0.140	157.76	0.910	72	4	-	-	-	-
	50	0.175	196.76	0.727	72	12	-	-	-	-
	51	0.188	234.82	0.959	69	12	-	-	-	-
Large framework (56 iterations)	42	0.139	75.53	0.440	27	28	5.940	0.979	1.973	0.915
	43	0.149	89.98	0.162	88	48	9.638	0.795	1.237	0.944
	44	0.128	81.45	0.312	18	28	9.742	0.700	2.380	0.939
	45	0.147	85.91	0.618	16	4	6.834	0.816	1.149	0.604
	46	0.122	84.28	0.037	4	28	5.820	0.997	2.817	0.141
	47	0.145	90.05	0.579	53	12	6.585	0.610	2.737	0.813
	48	0.139	79.56	0.452	29	4	4.035	0.483	2.752	0.955
	49	0.128	81.67	0.126	7	28	7.151	0.387	2.897	0.505
	50	0.150	82.38	0.204	35	28	4.785	0.663	2.449	0.744
	51	0.134	88.64	0.015	3	80	6.632	0.685	1.381	0.531

Table A-5: Random search: exploring the parameter space purely random for different framework sizes. The displayed values are the best found configurations during the allowable amount of iterations (differs per framework). The `random_state` is varied to achieve randomized, but reproducible results.

	random_state	Final CE	Runtime (min)	Final parameter configuration						
				spatial_lambda	n_neighbors	n_neighbors_spatial	learning_rate	min_dist	spread	set_op_mix_ratio
Small framework (3 random, 5 intelligent iterations)	42	0.174	4.26	0.672	-	-	-	-	-	-
	43	0.169	4.08	0.721	-	-	-	-	-	-
	44	0.170	4.06	0.716	-	-	-	-	-	-
	45	0.172	4.05	0.675	-	-	-	-	-	-
	46	0.170	4.01	0.696	-	-	-	-	-	-
	47	0.170	4.00	0.673	-	-	-	-	-	-
	48	0.171	4.09	0.697	-	-	-	-	-	-
	49	0.171	4.09	0.750	-	-	-	-	-	-
	50	0.170	4.11	0.706	-	-	-	-	-	-
	51	0.170	3.90	0.710	-	-	-	-	-	-
Medium framework (9 random, 15 intelligent iterations)	42	0.202	12.95	0.601	46	12	-	-	-	-
	43	0.181	30.81	0.502	12	12	-	-	-	-
	44	0.165	37.23	0.856	67	12	-	-	-	-
	45	0.148	37.18	0.805	23	4	-	-	-	-
	46	0.259	41.98	0.237	20	112	-	-	-	-
	47	0.173	33.11	0.690	27	12	-	-	-	-
	48	0.143	31.29	0.961	82	4	-	-	-	-
	49	0.158	31.16	0.777	43	4	-	-	-	-
	50	0.165	33.45	0.736	32	4	-	-	-	-
	51	0.142	30.67	0.963	99	4	-	-	-	-
Large framework (21 random, 35 intelligent iterations)	42	0.103	70.73	0.010	11	4	10.0	1.0	3.0	0.667
	43	0.102	75.46	0.010	13	28	5.846	1.0	2.057	1.0
	44	0.108	68.10	0.010	8	4	0.1	1.0	3.0	1.0
	45	0.131	81.97	0.803	100	4	10.0	1.0	1.751	1.0
	46	0.122	81.10	0.037	4	28	5.82	0.997	2.817	0.141
	47	0.101	81.64	0.338	12	4	10.00	1.0	3.0	1.0
	48	0.103	86.62	0.010	11	196	0.574	1.0	2.911	0.876
	49	0.099	91.15	0.059	9	12	0.376	1.0	2.476	1.0
	50	0.130	78.22	0.010	37	112	9.148	1.0	3.0	0.823
	51	0.132	83.10	0.761	84	4	4.594	1.0	2.614	1.0

Table A-6: Bayesian optimization: exploring the parameter following the Bayesian optimization method, for different framework sizes. The displayed values are the best found configurations during the allowable amount of iterations (differs per framework). The `random_state` is varied to achieve randomized, but reproducible results.

Bibliography

[1] A. R. Buchberger, K. DeLaney, J. Johnson, and L. Li, "Mass Spectrometry Imaging: A Review of Emerging Advancements and Future Insights," *Analytical Chemistry*, vol. 90, no. 1, pp. 240–265, 2018. [Online]. Available: https://doi.org/10.1021/acs.analchem.7b04733

[2] T. Alexandrov and J. H. Kobarg, "Efficient spatial segmentation of large imaging mass spectrometry datasets with spatially aware clustering," *Bioinformatics*, vol. 27, no. 13, pp. 230–238, 2011. [Online]. Available: http://doi.org/10.1093/bioinformatics/btr246

[3] N. Verbeeck, R. M. Caprioli, and R. V. D. Plas, "Unsupervised Machine Learning for Exploratory Data Analysis in Imaging Mass Spectometry," *Mass Spectrometry Reviews*, vol. 39, no. 3, pp. 245–291, 2020. [Online]. Available: https://doi.org/10.1002/mas.21602

[4] R. Van de Plas, "Computational aspects," Nashville, TN, 2018.

[5] I. Jolliffe, *Principal Component Analysis*, 2nd ed. New York, NY: Springer-Verlag New York, Inc., 2002. [Online]. Available: https://doi.org/10.1007/b98835

[6] L. V. D. Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[7] G. Hinton and S. Roweis, "Stochastic neighbor embedding," in *Advances in Neural Information Processing Systems*, 2003.

[8] L. Van der Maaten, "Barnes-Hut-SNE," in *1st International Conference on Learning Representations, ICLR 2013 - Conference Track Proceedings*, 2013. [Online]. Available: https://arxiv.org/abs/1301.3342

[9] G. C. Linderman, M. Rachh, J. G. Hoskins, S. Steinerberger, and Y. Kluger, "Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data," *Nature Methods*, vol. 16, no. 3, pp. 243–245, 2019. [Online]. Available: http://doi.org/10.1038/s41592-018-0308-4

[10] G. C. Linderman and S. Steinerberger, "Clustering with t-sne, provably." 2017. [Online]. Available: https://doi.org/10.1137/18m1216134

[11] L. Mcinnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," 2018. [Online]. Available: http://arxiv.org/abs/1802.03426

[12] A. C. Belkina, C. O. Ciccolella, R. Anno, R. Halpert, J. Spidlen, and J. E. Snyder-Cappione, "Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets," *Nature Communications*, vol. 10, no. 1, 2019. [Online]. Available: http://doi.org/10.1038/s41467-019-13055-y

[13] S. S. Rubakhin, J. C. Jurchen, E. B. Monroe, and J. V. Sweedler, "Imaging mass spectrometry: Fundamentals and applications to drug discovery," *Drug Discovery Today*, vol. 10, no. 12, pp. 823–837, 2005. [Online]. Available: https://doi.org/10.1016/S1359-6446(05)03458-6

[14] J. G. Swales, G. Hamm, M. R. Clench, and R. J. Goodwin, "Mass spectrometry imaging and its application in pharmaceutical research and development: A concise review," *International Journal of Mass Spectrometry*, vol. 437, pp. 99–112, 2019. [Online]. Available: https://doi.org/10.1016/j.ijms.2018.02.007

[15] A. F. M. Altelaar, L. A. Mcdonnell, S. R. Piersma, and R. M. A. Heeren, "Imaging mass spectrometry at cellular length scales," *Nature Protocols*, vol. 2, no. 5, pp. 1185–1196, 2007. [Online]. Available: http://doi.org/10.1038/nprot.2007.117

[16] S. A. Schwartz, M. L. Reyzer, and R. M. Caprioli, "Direct tissue analysis using matrix-assisted laser desorption/ionization mass spectrometry: practical aspects of sample preparation," *Journal of Mass Spectrometry*, vol. 38, pp. 699–708, 2003. [Online]. Available: https://doi.org/10.1002/jms.505

[17] C. L. Carter, J. W. Jones, A. M. Farese, T. J. Macvittie, and M. A. Kane, "Inflation-Fixation Method for Lipidomic Mapping of Lung Biopsies by Matrix Assisted Laser Desorption/Ionization - Mass Spectrometry Imaging," *Analytical Chemistry*, vol. 88, no. 9, pp. 4788–4794, 2016. [Online]. Available: https://doi.org/10.1021/acs.analchem.6b00165

[18] M. Gessel, J. M. Spraggins, P. Voziyan, G. Hudson, and R. M. Caprioli, "Decellularization of intact tissue enables MALDI imaging mass spectrometry analysis of the extracellular matrix," *Journal of Mass Spectometry*, vol. 50, no. 11, pp. 1288–1293, 2015. [Online]. Available: https://doi.org/10.1002/jms.3696

[19] E. R. Amstalden van Hove, D. F. Smith, and R. M. Heeren, "A concise review of mass spectrometry imaging," *Journal of Chromatography A*, vol. 1217, no. 25, pp. 3946–3954, 2010. [Online]. Available: http://doi.org/10.1016/j.chroma.2010.01.033

[20] J. Oetjen, K. Veselkov, J. Watrous, J. S. McKenzie, M. Becker, L. Hauberg-Lotte, J. H. Kobarg, N. Strittmatter, A. K. Mróz, F. Hoffmann, D. Trede, A. Palmer, S. Schiffler, K. Steinhorst, M. Aichler, R. Goldin, O. Guntinas-Lichius, F. v. Eggeling, H. Thiele, K. Maedler, A. Walch, P. Maass, P. C. Dorrestein,

Z. Takats, and T. Alexandrov, "Benchmark datasets for 3D MALDI- and DESI-imaging mass spectrometry," *GigaScience*, vol. 4, no. 1, 2015. [Online]. Available: https://doi.org/10.1186/s13742-015-0059-4

[21] Z. Takáts, J. M. Wiseman, B. Gologan, and R. G. Cooks, "Mass Spectrometry Sampling Under Ambient Conditions with Desorption Electrospray Ionization," *Science*, vol. 306, no. 5695, pp. 471–474, 2004. [Online]. Available: https://doi.org/10.1126/science.1104404

[22] J. M. Wiseman, D. R. Ifa, Q. Song, and R. G. Cooks, "Tissue Imaging at Atmospheric Pressure Using Desorption Electrospray Ionization (DESI) Mass Spectrometry," *Angewandte Chemie - International Edition*, vol. 45, no. 43, pp. 7188–7192, 2006. [Online]. Available: https://doi.org/10.1002/anie.200602449

[23] F. M. Green, T. L. Salter, I. S. Gilmore, P. Stokes, and G. O. Connor, "The effect of electrospray solvent composition on desorption electrospray ionisation (DESI) efficiency and spatial resolution," *Analyst*, vol. 135, pp. 731–737, 2010. [Online]. Available: https://doi.org/10.1039/b924208b

[24] S. Chandra, D. R. Smith, and G. H. Morrison, "A Subcellular Imaging by Dynamic SIMS Ion Microscopy," *Analytical Chemistry*, vol. 72, no. 3, pp. 104 A–114 A, 2000. [Online]. Available: https://doi.org/10.1021/ac002716i

[25] R. M. Caprioli, T. B. Farmer, and J. Gile, "Molecular Imaging of Biological Samples : Localization of Peptides and Proteins Using MALDI-TOF MS," *Analytical Chemistry*, vol. 69, no. 23, pp. 4751–4760, 1997. [Online]. Available: https://doi.org/10.1021/ac970888i

[26] H. Bai, S. Wang, J. Liu, D. Gao, Y. Jiang, H. Liu, and Z. Cai, "Localization of ginsenosides in Panax ginseng with different age by matrix-assisted laser-desorption/ionization time-of-flight mass spectrometry imaging," *Journal of Chromatography B: Analytical Technologies in the Biomedical and Life Sciences*, vol. 1026, pp. 263–271, 2016. [Online]. Available: http://doi.org/10.1016/j.jchromb.2015.09.024

[27] M. Stoeckli, P. Chaurand, D. E. Hallahan, and R. M. Caprioli, "Imaging mass spectrometry: A new technology for the analysis of protein expression in mammalian tissues," *Nature Medicine*, vol. 7, no. 4, pp. 493–496, 2001. [Online]. Available: https://doi.org/10.1038/86573

[28] T. Alexandrov, "MALDI imaging mass spectrometry: statistical data analysis and current computational challenges." *BMC Bioinformatics*, vol. 13, no. Suppl 16, 2012. [Online]. Available: http://doi.org/10.1186/1471-2105-13-s16-s11

[29] M. Aichler and A. Walch, "MALDI Imaging mass spectrometry: Current frontiers and perspectives in pathology research and practice," *Laboratory Investigation*, vol. 95, no. 4, pp. 422–431, 2015. [Online]. Available: http://doi.org/10.1038/labinvest.2014.156

[30] B. Balluff, M. Elsner, A. Kowarsch, S. Rauser, S. Meding, C. Schuhmacher, M. Feith, K. Herrmann, C. Ro, R. M. Schmid, H. Ho, A. Walch, and M. P. Ebert, "Classification of HER2/neu Status in Gastric Cancer Using a Breast-Cancer Derived Proteome Classifier research articles," *Journal of Proteome Research*, vol. 9, no. 12, pp. 6317–6322, 2010. [Online]. Available: https://doi.org/10.1021/pr100573s

[31] K. Schwamborn, M. Kriegsmann, and W. Weichert, "MALDI imaging mass spectrometry — From bench to bedside," *Biochimica et Biophysica Acta*, vol. 1865, no. 7, pp. 776–783, 2017. [Online]. Available: http://doi.org/10.1016/j.bbapap.2016.10.014

[32] G. Chen, B. Ning, and T. Shi, "Single-cell RNA-seq technologies and related computational data analysis," *Frontiers in Genetics*, vol. 10, no. APR, pp. 1–13, 2019. [Online]. Available: https://doi.org/10.3389/fgene.2019.00317

[33] L. A. Mcdonnell and R. M. Heeren, "Imaging Mass Spectrometry," *Mass Spectrometry Reviews*, vol. 26, pp. 606–643, 200. [Online]. Available: https://doi.org/10.1002/mas.20124

[34] W. Wiley and I. McLaren, "Time-of-Flight Mass Spectrometer with Improved Resolution," *Review of Scientific Instruments*, vol. 26, no. 12, pp. 1150–1157, 1955. [Online]. Available: https://doi.org/10.1063/1.1715212

[35] S. S. Rubakhin and J. V. Sweedler, "A mass spectrometry primer for mass spectrometry imaging," *Methods Mol Biol.*, vol. 656, no. 1, pp. 21–49, 2010. [Online]. Available: https://doi.org/10.1007/978-1-60761-746-4_2

[36] E. A. Jones, S.-o. Deininger, P. C. W. Hogendoorn, A. M. Deelder, and L. A. Mcdonnell, "Imaging mass spectrometry statistical analysis," *Journal of Proteomics*, vol. 75, no. 16, pp. 4962–4989, 2012. [Online]. Available: http://doi.org/10.1016/j.jprot.2012.06.014

[37] J. L. Norris, D. S. Cornett, J. A. Mobley, M. Andersson, E. H. Seeley, P. Chaurand, and R. M. Caprioli, "Processing MALDI mass spectra to improve mass spectral direct tissue analysis," *International Journal of Mass Spectrometry*, vol. 260, no. 2-3, pp. 212–221, 2007. [Online]. Available: https://doi.org/10.1016/j.ijms.2006.10.005

[38] W. E. Wolski, M. Lalowski, P. Martus, R. Herwig, P. Giavalisco, J. Gobom, A. Sickmann, H. Lehrach, and K. Reinert, "Transformation and other factors of the peptide mass spectrometry pairwise peak-list comparison process," *BMC Bioinformatics*, vol. 6, 2005. [Online]. Available: https://doi.org/10.1186/1471-2105-6-285

[39] J. Jackson, *A user's guide to principal components*. Hoboken, NJ: John Wiley & Sons, Inc, 1991.

[40] M. R. Keenan and P. G. Kotula, "Accounting for Poisson noise in the multivariate analysis of ToF-SIMS spectrum images," *Surface and Interface Analysis*, vol. 36, no. 3, pp. 203–212, 2004. [Online]. Available: https://doi.org/10.1002/sia.1657

[41] S.-o. Deininger, D. S. Cornett, R. Paape, M. Becker, C. Pineau, S. Rauser, A. Walch, and E. Wolski, "Normalization in MALDI-TOF imaging datasets of proteins: practical considerations," *Analytical and Bioanalytical Chemistry*, vol. 401, no. 1, pp. 167–181, 2011. [Online]. Available: https://doi.org/10.1007/s00216-011-4929-z

[42] C. A. Vallejos, D. Risso, A. Scialdone, S. Dudoit, and J. C. Marioni, "Normalizing single-cell RNA sequencing data : challenges and opportunities," *Nature Methods*, vol. 14, no. 6, pp. 565–571, 2017. [Online]. Available: https://doi.org/10.1038/nmeth.4292

[43] M. B. Tracy, H. Chen, D. M. Weaver, D. I. Malyarenko, M. Sasinowski, L. H. Cazares, R. R. Drake, O. J. Semmes, E. R. Tracy, and W. E. Cooke,

"Precision enhancement of MALDI-TOF MS using high resolution peak detection and label-free alignment," *Proteomics*, vol. 8, pp. 1530–1538, 2008. [Online]. Available: https://doi.org/10.1002/pmic.200701146

[44] A. D. Palmer, J. Bunch, and I. B. Styles, "Randomized approximation methods for the efficient compression and analysis of hyperspectral data," *Analytical Chemistry*, vol. 85, no. 10, pp. 5078–5086, 2013. [Online]. Available: https://doi.org/10.1021/ac400184g

[45] C. Bauer, R. Cramer, and J. Schuchhardt, "Evaluation of Peak-Picking Algorithms for Protein Mass Spectrometry," in *Data Mining in Proteomics*, M. Hamacher, M. Eisenacher, and C. Stephan, Eds., Clifton, N.J., 2011, vol. 696, pp. 341–352. [Online]. Available: https://doi.org/10.1007/978-1-60761-987-1_22

[46] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers and Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014. [Online]. Available: http://doi.org/10.1016/j.compeleceng.2013.11.024

[47] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003. [Online]. Available: https://doi.org/10.1162/089976603321780317

[48] I. Guyon and A. Elisseef, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[49] T. S. Andrews and M. Hemberg, "Identifying cell populations with scRNASeq," *Molecular Aspects of Medicine*, vol. 59, pp. 114–122, 2018. [Online]. Available: https://doi.org/10.1016/j.mam.2017.07.002

[50] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the Surprising Behavior of Distance Metrics in High Dimensional Space," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics; 1973)*. Berlin: Springer, 2001, pp. 420–434. [Online]. Available: https://doi.org/10.1007/3-540-44503-x_27

[51] S. Theodoridis and K. Koutroumbas, "Classifiers Based on Bayes Decision Theory 2 2.1," in *Pattern Recognition*, 4th ed., 2009, ch. 2, pp. 13–89. [Online]. Available: https://doi.org/10.1016/B978-1-59749-272-0.50004-9

[52] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, no. 6, pp. 417–441, 1933. [Online]. Available: https://doi.org/10.1037/h0071325

[53] C. Jutten and J. Herault, "Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture," *Signal Processing*, vol. 24, no. 1, pp. 1–10, 1991. [Online]. Available: https://doi.org/10.1016/0165-1684(91)90079-X

[54] P. Switzer and A. Green, "Min/max autocorrelation factors for multivariate spatial imagery," Stanford University, Stanford, California, Tech. Rep., 1984.

[55] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999. [Online]. Available: https://doi.org/10.1038/44565

[56] A. Narayan, B. Berger, and H. Cho, "Density-Preserving Data Visualization Unveils Dynamic Patterns of Single-Cell Transcriptomic Variability," 2020. [Online]. Available: https://doi.org/10.1101/2020.05.12.077776

[57] K. Pal and M. Sharma, "Performance evaluation of non-linear techniques UMAP and t-SNE for data in higher dimensional topological space," in *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Palladam, India, 2020, pp. 1106–1110. [Online]. Available: https://doi.org/10.1109/I-SMAC49090.2020.9243502

[58] B. M. Wilamowski and J. D. Irwin, *Intelligent Systems*, 2nd ed. Boca Raton, FL: CRC Press, Inc, 2011.

[59] P. R. Peres-neto, D. A. Jackson, and K. M. Somers, "How many principal components? Stopping rules for determining the number of non-trivial axes revisited," *Computational Statistics and Data Analysis*, vol. 49, no. 4, pp. 974–997, 2005. [Online]. Available: https://doi.org/10.1016/j.csda.2004.06.015

[60] S.-O. Deiniger, M. P. Ebert, A. Fu, and M. Gerhard, "MALDI Imaging Combined with Hierarchical Clustering as a New Tool for the Interpretation of Complex Human Cancers," *Journal of Proteome Reasearch*, vol. 7, no. 12, pp. 5230–5236, 2008. [Online]. Available: https://doi.org/10.1021/pr8005777

[61] M. Buscema, F. Vernieri, G. Massini, F. Scrascia, M. Breda, P. M. Rossini, and E. Grossi, "An improved I-FAST system for the diagnosis of Alzheimer's disease from unprocessed electroencephalograms by using robust invariant features," *Artificial Intelligence in Medicine*, vol. 64, no. 1, pp. 59–74, 2015. [Online]. Available: http://doi.org/10.1016/j.artmed.2015.03.003

[62] S. T. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *IEEE Transactions on Neural Networks*, vol. 290, no. 5500, pp. 2323–2326, 2000. [Online]. Available: https://doi.org/10.1126/science.290.5500.2323

[63] J. W. Sammon, "Nonlinear Mapping Structure Analysis," *IEEE Transactions on Computers*, vol. C-18, no. 5, pp. 401–409, 1969. [Online]. Available: https://doi.org/10.1109/T-C.1969.222678

[64] J. Kruskal, "Nonmetric Multidimensional Scaling: A Numerical Method," *Psychometrika*, vol. 29, no. 2, pp. 115–129, 1964. [Online]. Available: https://doi.org/10.1007/BF02289694

[65] T. Kohonen, "The Self-organizing Map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990. [Online]. Available: https://doi.org/10.1016/S0925-2312(98)00030-7

[66] R. R. Coifman and S. Lafon, "Diffusion maps," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 5–30, 2006. [Online]. Available: https://doi.org/10.1016/j.acha.2006.04.006

[67] L. Van Der Maaten, "Accelerating t-SNE using tree-based algorithms," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3221–3245, 2014.

[68] J. Tang, J. Liu, M. Zhang, and Q. Mei, "Visualizing Large-scale and High-dimensional Data," in *Association for Computing Machinery (ACM)*, 2016, pp. 287–297. [Online]. Available: https://doi.org/10.1145/2872427.2883041

[69] A. Artemenkov and M. Panov, "NCVis: Noise Contrastive Approach for Scalable Visualization," in *The Web Conference 2020 - Proceedings of the World Wide Web Conference, WWW 2020*, 2020, pp. 2941–2947. [Online]. Available: https://doi.org/10.1145/3366423.3380061

[70] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale Information Network Embedding Categories and Subject Descriptors," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077. [Online]. Available: https://doi.org/10.1145/2736277.2741093

[71] J. Ding, A. Condon, and S. P. Shah, "Interpretable dimensionality reduction of single cell transcriptome data with deep generative models," *Nature Communications*, vol. 9, no. 1, 2018. [Online]. Available: http://doi.org/10.1038/s41467-018-04368-5

[72] J. Cook, I. Sutskever, A. Mnih, and G. Hinton, "Visualizing Similarity Data with a Mixture of Maps," in *Journal of Machine Learning Research*, vol. 2, 2007, pp. 67–74.

[73] N. Oskolkov, "How Exactly UMAP Works," 2019. [Online]. Available: https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668

[74] J. Barnes and P. Hut, "A hierarchical O(N log N) force-calculation algorithm," *Nature*, vol. 324, no. 4, pp. 446–449, 1986. [Online]. Available: https://doi.org/10.1038/324446a0

[75] A. G. Gray and A. W. Moore, "'N-Body' problems in statistical learning," in *Advances in Neural Information Processing Systems*, 2001.

[76] N. Pezzotti, T. Höllt, B. Lelieveldt, E. Eisemann, and A. Vilanova, "Hierarchical Stochastic Neighbor Embedding," *Computer Graphics Forum*, vol. 35, no. 3, pp. 21–30, 2016. [Online]. Available: https://doi.org/10.1111/cgf.12878

[77] V. V. Unen, T. Höllt, N. Pezzotti, N. Li, M. J. T. Reinders, F. Koning, A. Vilanova, B. P. F. Lelieveldt, and E. Eisemann, "Visual analysis of mass cytometry data by hierarchical stochastic neighbour embedding reveals rare cell types," *Nature Communications*, vol. 8, no. 1, 2017. [Online]. Available: https://doi.org/10.1038/s41467-017-01689-9

[78] T. Smets, N. Verbeeck, M. Claesen, A. Asperger, G. Griffioen, T. Tousseyn, W. Waelput, E. Waelkens, and B. De Moor, "Evaluation of Distance Metrics and Spatial Autocorrelation in Uniform Manifold Approximation and Projection Applied to Mass Spectrometry Imaging Data," *Analytical Chemistry*, vol. 91, no. 9, pp. 5706–5714, 2019. [Online]. Available: https://doi.org/10.1021/acs.analchem.8b05827

[79] L. J. Winderbaum, I. Koch, O. J. R. Gustafsson, S. Meding, and P. Hoffmann, "Feature extraction for proteomics imaging mass spectrometry data," *Annals of Applied Statistics*, vol. 9, no. 4, pp. 1973–1996, 2015. [Online]. Available: https://doi.org/10.1214/15-AOAS870

[80] J. Cao, M. Spielmann, X. Qiu, X. Huang, D. M. Ibrahim, A. J. Hill, F. Zhang, S. Mundlos, L. Christiansen, F. J. Steemers, C. Trapnell, and J. Shendure, "The single-cell transcriptional landscape of mammalian organogenesis," *Nature*, vol. 566, no. 7745, pp. 496–502, 2019. [Online]. Available: http://doi.org/10.1038/s41586-019-0969-x

[81] G. Qian, S. Sural, Y. Gu, and S. Pramanik, "Similarity between euclidean and cosine angle distance for nearest neighbor queries," in *Proceedings of the ACM Symposium on Applied Computing*, vol. 2, 2004, pp. 1232–1237. [Online]. Available: https://doi.org/10.1145/967900.968151

[82] B. Desgraupes, "Clustering Indices," 2017. [Online]. Available: https://cran.r-project.org/web/packages/clusterCrit/vignettes/clusterCrit.pdf

[83] S. Sarkari, C. D. Kaddi, R. V. Bennett, F. M. Fernandez, and M. D. Wang, "Comparison of clustering pipelines for the analysis of mass spectrometry imaging data," in *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2014*, 2014, pp. 4771–4774. [Online]. Available: https://doi.org/10.1109/EMBC.2014.6944691

[84] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 10, no. 1, 2015. [Online]. Available: https://doi.org/10.1145/2733381

[85] T. Sainburg, L. Mcinnes, and T. Q. Gentner, "Parametric UMAP: learning embeddings with deep neural networks for representation and semi-supervised learning," 2020. [Online]. Available: https://arxiv.org/abs/2009.12981v2

[86] W. Dong, M. Charikar, and K. Li, "Efficient K-Nearest Neighbor Graph Construction for Generic Similarity Measures," in *Proceedings of the 20th International Conference on World Wide Web, WWW 2011*, 2011, pp. 577–586. [Online]. Available: https://doi.org/10.1145/1963405.1963487

[87] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975. [Online]. Available: https://doi.org/10.1145/361002.361007

[88] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An Algorithm for Finding Best Matches in Logarithmic Expected Time," *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, no. 3, pp. 209–226, 1977. [Online]. Available: https://doi.org/10.1145/355744.355745

[89] P. N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," in *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*. ACM, 1993, pp. 311–321.

[90] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *ACM International Conference Proceeding Series*, vol. 148, 2006, pp. 97–104. [Online]. Available: http://doi.org/10.1145/1143844.1143857

[91] S. Dasgupta and Y. Freund, "Random projection trees and low dimensional manifolds," in *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2008, pp. 537–546. [Online]. Available: https://doi.org/10.1145/1374376.1374452

[92] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *27th Annual Conference on Neural Information Processing Systems, NIPS 2013*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., Lake Tahoe, NV, 2013. [Online]. Available: https://arxiv.org/abs/1606.08359

[93] Y. Goldberg and O. Levy, "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method," 2014. [Online]. Available: http://arxiv.org/abs/1402.3722

[94] F. Niu, B. Recht, C. Ré, and S. J. Wright, "HOGWILD!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent," in *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*, 211. [Online]. Available: https://arxiv.org/abs/1106.5730

[95] C. J. Nolet, V. Lafargue, E. Raff, T. Nanditale, T. Oates, J. Zedlewski, and J. Patterson, "Bringing UMAP Closer to the Speed of Light with GPU Acceleration," 2020. [Online]. Available: http://arxiv.org/abs/2008.00325

[96] D. Kobak and G. C. Linderman, "Initialization is critical for preserving global data structure in both t -SNE and UMAP," *Nature Biotechnology*, vol. 39, no. February, pp. 156–157, 2021. [Online]. Available: http://doi.org/10.1038/s41587-020-00809-z

[97] D. Kobak and P. Berens, "The art of using t-SNE for single-cell transcriptomics," *Nature Communications*, vol. 10, no. 1, 2019. [Online]. Available: http://doi.org/10.1038/s41467-019-13056-x

[98] T. Smets, E. Waelkens, and B. D. Moor, "Prioritization of m/z-Values in Mass Spectrometry Imaging Profiles Obtained Using Uniform Manifold Approximation and Projection for Dimensionality Reduction," *Analytical Chemistry*, vol. 92, no. 7, pp. 5240–5248, 2020. [Online]. Available: https://doi.org/10.1021/acs.analchem.9b05764

[99] M. W. Dorrity, L. M. Saunders, C. Queitsch, S. Fields, and C. Trapnell, "Dimensionality reduction by UMAP to visualize physical and genetic interactions," *Nature Communications*, vol. 11, no. 1, 2020. [Online]. Available: https://doi.org/10.1038/s41467-020-15351-4

[100] E. Becht, L. Mcinnes, J. Healy, C.-a. Dutertre, I. W. H. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell, "Dimensionality reduction for visualizing single-cell data using UMAP," *Nature Biotechnology*, vol. 37, no. 1, pp. 38–47, 2019. [Online]. Available: https://doi.org/10.1038/nbt.4314

[101] J. S. Packer, Q. Zhu, C. Huynh, P. Sivaramakrishnan, E. Preston, H. Dueck, D. Stefanik, K. Tan, C. Trapnell, J. Kim, R. H. Waterston, and J. I. Murray, "A lineage-resolved molecular atlas of C. elegans embryogenesis at single-cell resolution," *Science*, vol. 365, no. 6459, 2019. [Online]. Available: https://doi.org/10.1126/science.aax1971

[102] F. O. Bagger, S. Kinalis, and N. Rapin, "BloodSpot: a database of healthy and malignant haematopoiesis updated with purified and single cell mRNA sequencing profiles," *Nucleic Acids Research*, vol. 47, no. D1, pp. D881–D885, 2019. [Online]. Available: https://doi.org/10.1093/nar/gky1076

[103] J.-E. Park, K. Polański, K. Meyer, and S. A. Teichmann, "Fast Batch Alignment of Single Cell Transcriptomes Unifies Multiple Mouse Cell Atlases into an Integrated Landscape," 2018. [Online]. Available: https://doi.org/10.1101/397042

[104] B. Kamiński, P. Pralat, and F. Théberge, "A Scalable Unsupervised Framework for Comparing Graph Embeddings," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12091 LNCS, 2020, pp. 52–67.

[105] A. Cassese, S. R. Ellis, N. Ogrinc, E. Burgermeister, M. Ebert, A. Walch, A. M. J. M. V. D. Maagdenberg, L. A. Mcdonnell, R. M. A. Heeren, and B. Ballu, "Spatial Autocorrelation in Mass Spectrometry Imaging," *Analytical Chemistry*, vol. 88, no. 11, pp. 5871–5878, 2016. [Online]. Available: https://doi.org/10.1021/acs.analchem.6b00672

[106] D. A. Griffith and Y. Chun, "Spatial Autocorrelation and Spatial Filtering," in *Handbook of Regional Science*, M. Fisher and P. Nijkamp, Eds. Springer-Verlag Berlin Heidelberg, 2014, pp. 1477–1507. [Online]. Available: https://doi.org/10.1007/978-3-642-23430-9_72

[107] P. Moran, "Notes on Continuous Stochastic Phenomena," *Biometrika*, vol. 37, no. 1-2, pp. 17–23, 1950. [Online]. Available: https://doi.org/10.1093/biomet/37.1-2.17

[108] A. Getis and J. K. Ord, "The Analysis of Spatial Association by Use of Distance Statistics," *Geographical Analysis*, vol. 24, no. 3, pp. 189–206, 1992. [Online]. Available: https://doi.org/10.1111/j.1538-4632.1992.tb00261.x

[109] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[110] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes, The Art of Scientific Computing*, 3rd ed. New York, USA: Cambridge University Press, 2007.

[111] R. Bellman, *Adaptive Control Processes: A Guided Tour*. New Jersey: Princeton University Press, 1961. [Online]. Available: https://doi.org/10.2307/3611672

[112] G. P. Way and C. S. Greene, "Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders," in *Pacific Symposium on Biocomputing*, 2018, pp. 80–91. [Online]. Available: https://doi.org/10.1142/9789813235533_0008

[113] L. Zappia, B. Phipson, and A. Oshlack, "Splatter: Simulation of single-cell RNA sequencing data," *Genome Biology*, vol. 18, no. 1, pp. 1–15, 2017. [Online]. Available: https://doi.org/10.1186/s13059-017-1305-0

[114] Q. Hu and C. S. Greene, "Parameter tuning is a key part of dimensionality reduction via deep variational autoencoders for single cell RNA transcriptomics," in

Pacific Symposium on Biocomputing, vol. 24, 2019, pp. 362–373. [Online]. Available: https://doi.org/10.1142/9789813279827_0033

[115] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. D. Freitas, "Taking the Human Out of the Loop : A Review of Bayesian Optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016. [Online]. Available: https://doi.org/10.1109/JPROC.2015.2494218

[116] E. Brochu, V. M. Cora, and N. de Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning," 2010. [Online]. Available: http://arxiv.org/abs/1012.2599

[117] P. I. Frazier, "A Tutorial on Bayesian Optimization," 2018.

[118] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*, 2011.

[119] R. Bardenet and B. Kégl, "Surrogating the surrogate: Accelerating Gaussian-process-based global optimization with a mixture cross-entropy algorithm," *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, pp. 55–62, 2010.

[120] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," in *Advances in Neural Information Processing Systems*, vol. 4, Lake Tahoe, NV, 2012, pp. 2951–2959.

[121] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: a Python library for model selection and hyperparameter optimization," *Computational Science and Discovery*, vol. 8, no. 1, 2015. [Online]. Available: https://doi.org/10.1088/1749-4699/8/1/014008

[122] M. Spraggins, K. V. Djambazova, E. S. Rivera, L. G. Migas, E. K. Neumann, A. Fuetterer, J. Suetering, N. Goedecke, A. Ly, R. V. D. Plas, and R. M. Caprioli, "High-Performance Molecular Imaging with MALDI Trapped Ion-Mobility Time-of-Flight (timsTOF) Mass Spectrometry," *Analytical Chemistry*, vol. 91, no. 22, pp. 14 552–14 560, 2019.

[123] K. V. Djambazova, D. R. Klein, L. G. Migas, E. K. Neumann, E. S. Rivera, R. V. D. Plas, R. M. Caprioli, and M. Spraggins, "Resolving the Complexity of Spatial Lipidomics Using MALDI TIMS Imaging Mass Spectrometry," *Analytical Chemistry*, vol. 92, no. 19, pp. 13 290–13 297, 2020.

[124] L. E. Tideman, L. G. Migas, K. V. Djambazova, N. H. Patterson, R. M. Caprioli, J. M. Spraggins, and R. Van de Plas, "Automated biomarker candidate discovery in imaging mass spectrometry data through spatially localized Shapley additive explanations," *Analytica Chimica Acta*, vol. 1177, p. 338522, 2021. [Online]. Available: https://doi.org/10.1016/j.aca.2021.338522

Glossary

List of Acronyms

IMS	Imaging Mass Spectrometry
TOF	Time-of-Flight
FTICR	Fourier transform ion cyclotron resonance
ICR	ion cyclotron resonance
TIMS	trapped ion mobility spectrometry
Q-TOF	Quadrupole Time of Flight
PMF	Peptide Mass Fingerprinting
ROI	region of interest
TIC	Total Ion Count
scRNA-seq	single-cell RNA sequencing
m/z	mass-to-charge ratio
FFPE	Formalin fixed paraffin-embedded
ECM	extracellular matrix
HV	high vacuum
UHV	ultra-high vacuum
MALDI	Matrix Assisted Laser Desorption/Ionisation
SMALDI	scanning microprobe MALDI
IR-MALDESI	infrared MALDESI
SALDI	surface-assisted laser desorption/ionization
DESI	Desorption electrospray ionization
SIMS	Secondary Ion Mass Spectrometry
PCA	Principal Component Analysis
PCs	principle components

SVD Singular Value Decomposition
MAF Maximum Autocorrelation Factorization
ICA Independent Component Analysis
NMF Non-negative Matrix Factorization
PMF Positive Matrix Factorization
UMAP Uniform Manifold Approximation and Projection
SNE Stochastic Neighbor Embedding
t-SNE t-Distributed Stochastic Neighbor Embedding
HSNE Hierarchical Stochastic Neighbor Embedding
MDS multidimensional scaling
LLE Locally Linear Embedding
SOMs Self-Organizing Maps
NCVis Noise Contrastive Approach for Scalable Visualization
DR Dimension Reduction
NLDR Nonlinear Dimensionality Reduction
HD High-Dimensional
LD Low-Dimensional
KLD Kullback–Leibler divergence
JSD Jensen–Shannon divergence
CE Cross-Entropy
GP Gaussian Processes
TPE Tree-structured Parzen Estimator
GD Gradient Descent
SGD Stochastic Gradient Descent
EI Expected Improvement
SMBO Sequential Model Based Optimisation
NEG Negative sampling
DBN Deep Belief Network
TS Thompson sampling
EI expected improvement
ES entropy search
CH Calinski-Harabasz
kNN k-Nearest Neighbor
CAR Gaussian Conditional Autoregressive
vp vantage point
RP Random Projection

NN nearest neighbour

BBKNN batch balanced k-nearest neighbours

VAE Variational autoencoders

DD-UMAP Data-Driven UMAP