DISS. ETH NO. 28721

Representation Learning for Dimensionality Reduction, Irregularly-Sampled Sequences and Graphs

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by
Max Horn
Master of Science,
Ruprecht-Karls-Universität Heidelberg

born on
16.07.1993

citizen of
Germany

accepted on the recommendation of
Prof. Dr. Karsten Borgwardt
Prof. Dr. Julia Vogt
Prof. Dr. Christian L. Müller

2023

Acknowledgments

First and foremost I want to express my gratitude to my advisor Prof. Dr. Karsten Borgwardt for allowing me to do my Ph.D. in his group. His support, scientific input, guidance, and mentorship shaped me as an academic researcher and grew me into the scientist I am today. I am deeply thankful for the opportunities he allowed me to pursue and for the freedom he gave me in exploring research directions that I am passionate about. I especially want to thank him for his support in my endeavors after my time in his group by allowing me to follow unique job opportunities prior to having submitted my thesis. Further, I want to thank Prof. Dr. Julia Vogt for being my secondary advisor. Additionally, I thank her and Prof. Dr. Christian Müller for being part of my thesis examination committee and Prof. Dr. Daniel Müller for chairing the examination.

Next, I would like to thank all collaborators and coauthors with whom I had the pleasure to work during my studies. Their collaborative spirit allowed me to work on many interesting topics and learn from their diverse experiences. In particular, I would like to thank Dr. Michael Moor, Dr. Caroline Weis, Leslie O'Bray, Edward De Brouwer, Dr. Nicolas Bennet and Drago Plecko with whom I coauthored some of the works in this thesis. Further, I would like to express my uttermost gratitude to my friend, mentor, and colleague Dr. Bastian Rieck, whose way of doing science was an inspiration to me and who was always open to exploring another crazy idea in one of our countless "skunkworks" projects. He introduced me to the concepts of topological data analysis and measure theory and even helped me get my general math knowledge up to speed in problem-driven sessions on the whiteboard. Further, I want to thank him for his support in the finalization of this thesis by proofreading and providing valuable feedback to *all* chapters. Without him, I would only be a shadow of the researcher I am today. Moreover, I want to thank my officemates Christian Bock and Michael Moor for their comradeship and close collaboration, which lead to many successful research projects. I want to thank Bastian, Christian, and Michael for the countless discussions we had over coffee, for their support during my Ph.D., and for having the honor to call them my friends.

I further want to express my gratitude to Dr. Damián Roqueiro, Matteo Togninalli, Dr. Laetitia Papaxanthos, Dr. Felipe Llinares-López, Dr. Thomas Gumbsch, Catherine Jutzeler, and Bowen Fan with whom I had the pleasure to collaborate. Additionally, I want to thank all past and present members of the MLCB lab including, Dr. Katharina Heinrich, Giulia Muzio, Tim Kucera, Lucie Bourguignon, Dr. Felix Hensel, Dr. Sarah Brüenigk, Dr. Dexiong Chen, Dr. Carlos Oliver, Dr. Michael Adamer, Dr. Elisabetta Ghisu, Dr. Anja Gumpinger, and Dr. Lukas Folkman for being wonderful and inspiring colleagues. Additionally, I would like to thank the staff of the BSSE in particular, Cindy Malnasi, Inga Pillinger and Dr. Andrea Huber Brösamle for supporting me in all organizational matters and matters with respect to the doctorate.

Finally, I would like to thank my family and friends for supporting me during this journey. In particular, I am deeply grateful to my father and mother who taught me to be fascinated by the world and who supported me in all my early pursuits of understanding computers and electronics which lead to many broken devices and an over years dysfunctional media center. I thank them for their unconditional love, for allowing me to choose my own path, and for supporting me in any halfway sensible endeavor. Further, I want to also thank Paul Gundel for the many years of sincere friendship and Chis Wendler for always having an open ear to listen to my woes. Lastly, I want to thank my incredibly wonderful wife Sophie. Her support, care, and love provided me with the energy to complete this journey which I would certainly not have made without her. I thank her for always being there for me and for always being a true friend and companion.

Abstract

Machine learning has the potential to revolutionize the fields of biology and healthcare by providing new tools to help scientists and clinicians do research and decide what would be the right treatment for patients. However, while recent approaches in representation learning give the impression of being universal black-box solutions to all problems, research has shown that this is not generally true. Even though models can perform well in a black-box fashion, they often suffer from low generalization and are sensitive to distribution shifts. This highlights the need for developing approaches that are informed by their downstream application and tailored to incorporate symmetries of the problem into the model architecture. These *inductive biases* are essential for performance on new data and for models to remain robust even when the data distribution changes. Nevertheless, constructing good models is only half of the solution. To be sure that models would translate well into clinical applications they also need to be evaluated appropriately with this goal in mind. In this thesis, I address the above points while taking a detailed look at structured data types present at the intersection of biology, medicine, and machine learning. In terms of algorithmic contributions, I first present a new non-linear dimensionality reduction algorithm that aims to preserve multi-scale relations. The cost reduction of genome sequencing and the ability to sequence individual cells has led to exponentially increasing high-dimensional data in the life sciences. Such data cannot be intuitively understood, making dimensionality reduction approaches, which can capture the nested relationships present in biology, essential. Second, I develop methods for clinical applications where irregularly-sampled data are present. Conventional machine learning models either require the conversion of such data into fixed-size representations or the imputation of missing values prior to their application. I present two approaches tailored for irregularly-sampled data that do not require such preprocessing steps. The first is a new kernel for peaks derived from MALDI-TOF spectra, whereas the second is a deep learning model that can be applied to irregularly-sampled time series by phrasing them as sets of observations. Third, I present an extension to graph neural networks that allow the models to account for global information instead of requiring nodes to only exchange information with their neighbors. Graphs are an important data structure for pharmacology as they are often used to represent small molecules. In order to address the appropriate evaluation of such models, I present a detailed study of medical time series models with a focus on their capability to transfer to other datasets in the context of a sepsis early prediction task. Further, I show that the conventional approach for the evaluation of graph generative models is highly sensitive to the selection of hyperparameters which can lead to biased performance estimates. Summarizing, my thesis addresses many problems at the intersection of machine learning, healthcare, and biology. It demonstrates how models can be improved by including more (domain-specific) knowledge and where to pay attention when evaluating said models.

Zusammenfassung

Maschinelles Lernen hat das Potenzial, Biologie und Medizin zu revolutionieren, indem es neue Werkzeuge bereitstellt, die Wissenschaftler und Kliniker bei der Forschung und bei der Entscheidung über die richtige Behandlung für Patienten unterstützen. Während die jüngsten Ansätze im Bereich des Repräsentationslernens den Eindruck erwecken, universelle Lösungen für alle Probleme zu sein, deutet aktuelle Forschung darauf hin, dass dies nicht generell der Fall ist. Auch wenn solche Modelle bei naiver Anwendung gut funktionieren können, leiden sie oft unter einer geringen Generalisierung und reagieren empfindlich auf Veränderungen der Datenverteilung. Daher ist es wichtig Ansätze zu entwickeln, die sich an der Anwendung orientieren und darauf zugeschnitten sind und so Symmetrien des Problems in die Modellarchitektur einbeziehen können. Diese "inductive biases" sind entscheidend für die Leistungsfähigkeit auf neuen Daten und dafür, dass die Modelle robust bleiben, auch wenn sich die Datenverteilung ändert. Dennoch ist die Konstruktion guter Modelle nur eine Hälfte der Lösung. Um sicher zu sein, dass sich Modelle gut in klinische Anwendungen übertragen lassen, müssen sie auch mit diesem Ziel vor Augen evaluiert werden. In dieser Arbeit adressiere ich die oben genannten Punkte und konzentriere mich dabei auf strukturierte Datentypen an der Schnittstelle zwischen Biologie, Medizin und maschinelles Lernen. Als algorithmischen Beitrag zum Feld stelle ich zuerst einen neuen Ansatz zur nichtlinearen Dimensionsreduktion vor, der darauf abzielt, Beziehungen auf mehreren Skalen zu erhalten. Die Kostenreduzierung bei der Genomsequenzierung und die Möglichkeit, einzelne Zellen zu sequenzieren, haben zu einem exponentiellen Anstieg hochdimensionalen Daten in den Biowissenschaften geführt. Solche Daten können nicht intuitiv verstanden werden, so dass Dimensionsreduktionsansätze die die verschachtelten Beziehungen in der Biologie erfassen können, unerlässlich sind. Zweitens, entwickle ich Modelle für klinische Anwendungen, bei denen unregelmäßig abgetastete Daten vorliegen. Konventionelle Modelle des maschinellen lernens erfordern entweder die Umwandlung solcher Daten in Repräsentationen fester Größe oder die Imputation fehlender Werte vor ihrer Anwendung. Ich stelle zwei Ansätze vor, die für unregelmäßig abgetastete Daten entwickelt wurden und keine solchen Vorverarbeitungsschritte erfordern. Der Erste ist ein neuer Kernel für Peaks aus MALDI-TOF-Spektren, während der Zweite ein Deep-Learning-Modell ist, das auf unregelmäßig abgetastete Zeitreihen angewendet werden kann, indem man sie als Mengen von Beobachtungen formuliert. Drittens, stelle ich eine Erweiterung von Graph Neural Networks vor, die es den Modellen ermöglicht, globale Informationen zu berücksichtigen, anstatt den Nachrichtenaustausch von Knoten auf ihre Nachbarschaft zu limitieren. Graphen sind eine wichtige Datenstruktur für die Pharmakologie, da sie häufig zur Darstellung von Molekülen verwendet werden. Um die korrekte Evaluierung solcher Modelle zu adressieren, präsentiere ich eine detailiere Studie medizinischer Zeitreihenmodelle mit Schwerpunkt auf deren Übertragbarkeit auf andere Datensätze im Kontext einer Sepsis-Frühwarnstudie.

Schließlich zeige ich, dass der konventionelle Ansatz zur Evaluation von generativen Graphenmodellen sehr empfindlich auf die Auswahl der Hyperparameter reagiert, was zu verzerrten Leistungseinschätzungen führen kann. Zusammengefasst befasst sich meine Arbeit mit vielen Problemen an der Schnittstelle von maschinellem Lernen, Medizin und Biologie. Sie zeigt, wie Modelle durch Einbeziehung von mehr domänenspezifischem Wissen verbessert werden können und worauf bei der Bewertung dieser Modelle zu achten ist.

Contents

Publications

The work presented in this thesis is based on the following publications:

- M. Moor*, M. Horn*, et al. "Topological Autoencoders". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7045–7054
- C. Weis*, M. Horn*, B. Rieck*, et al. "Topological and kernel-based microbial phenotype prediction from MALDI-TOF mass spectra". *Bioinformatics* 36, 2020, pp. i30–i38
- M. Horn et al. "Set functions for time series". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 4353–4363
- M. Moor*, N. Bennet*, D. Plecko*, M. Horn*, et al. "Predicting sepsis in multi-site, multi-national intensive care cohorts using deep learning". Preprint. 2021. arXiv: 2107.05230 [cs.LG]
- M. Horn*, E. D. Brouwer*, et al. "Topological Graph Neural Networks". In: *International Conference on Learning Representations*. 2022
- L. O'Bray*, M. Horn*, et al. "Evaluation Metrics for Graph Generative Models: Problems, Pitfalls, and Practical Solutions". In: *International Conference on Learning Representations*. 2022

The corresponding chapters additionally refer to the works above in their introduction. Besides the publications presented in this thesis I contributed to the following works during my doctoral studies:

- M. Horn*, K. Shridhar*, E. Groenewald*, and P. F. Baumann*. "Translational Equivariance in Kernelizable Attention". Preprint. 2021. arXiv: 2102.07680 [cs.LG]
- B. Rieck et al. "Neural Persistence: A Complexity Measure for Deep Neural Networks Using Algebraic Topology". In: *International Conference on Learning Representations*. 2019
- M. Moor et al. "Early recognition of sepsis with Gaussian process temporal convolutional networks and dynamic time warping". In: *Machine Learning for Healthcare Conference*. PMLR. 2019, pp. 2–26
- S. L. Hyland et al. "Early prediction of circulatory failure in the intensive care unit using machine learning". *Nature medicine* 26:3, 2020, pp. 364–373

- M. Moor et al. “Path Imputation Strategies for Signature Models”. In: *ICML Workshop on the Art of Learning with Missing Values (Artemiss)*. 2020
- M. Moor et al. “Challenging euclidean topological autoencoders”. In: *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*. 2020
- M. Moor et al. “Early prediction of sepsis in the ICU using machine learning: a systematic review”. *Frontiers in medicine* 8, 2021, p. 607952

1 Introduction

Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain. — Alan Turing

The field of representation learning (often also called *deep learning*) has shifted our expectations on what to expect from machine learning in everyday tasks. Tasks that seemed impossible a few years prior can now be solved by algorithms reliably and often better than human baselines [63]. After defeating humans in the game of chess [37], algorithms have moved on to reach superhuman performance in the game of Go [220] and also started to have an impact in the medical domain, where they compete with experts in skin cancer recognition performance [71]. While tackling chess can mostly be attributed to the machine's high computational performance and the ability to enumerate every possible opponent move, many tasks cannot be solved using a complete enumeration strategy. Recent advances in machine learning have managed to make progress in tasks where the number of potential combinations is larger than the number of atoms in the universe. Tasks ranging from recognizing objects in images [126] that we humans typically solve intuitively or without thinking after sufficient experience to predicting the folding of proteins [116] which we have not been able to reliably tackle despite over 60 years of research in this area.

While enabling an algorithm to recognize a cat in an image or understand the approximate content of a video is great for generating revenue by optimizing ad placement, only a few approaches have been designed and applied to improve the living standards of humans. One exception represents the application of machine learning algorithms to healthcare and biology. Machine learning algorithms are powerful tools for exploring high-dimensional data in research, predicting the properties of molecules in drug design or monitoring a patient's state in the intensive care unit. While some progress has been made towards transferring innovations to these fields, the data is often incompatible: Dimensionality reduction algorithms for biology should preserve distances across multiple scales [44], medical time-series show patterns of missingness due to irregular sampling [108, 162], models on graphs often fail to capture the global structure [34] and correctly evaluating generated small molecules is inherently difficult [174].

This thesis strives to fill some of the gaps that need to be filled for machine learning algorithms to find wider applications in the fields of biology and healthcare. Yet, it is *not* a comprehensive body of work that fixes all of them and claiming to would undermine the complexity of truly bringing these models into application. Instead, we here focus on designing models which can better handle the types of data that are present in biology and healthcare and highlight how evaluation of such methods needs to take place for performance estimates to hopefully translate to the real world.

Why Representation Learning? Much of machine learning research at the intersection with biology and healthcare has concentrated on incorporating as much expert knowledge as possible into the representations used by models. This process called *feature engineering* was mostly due to the limited availability of data. The main issue with this approach is that if the features are not selected appropriately, the performance of the method is upper bounded, and finding the right feature representation for the problem using manual (human-driven) search is highly non-trivial. Often it already requires knowing to some degree what would be needed in order to solve the task at hand. Problems begin to arise when we *do not know* which features to use because the problem is insufficiently explored or there are too many levels of complexity between what we can measure and the outcome we are trying to predict. In representation learning, task-specific features are learned from raw data with little to no human interaction through optimization[1].

Let us now look at some motivational context to highlight, why representation learning can be especially relevant for biology and healthcare. Due to the lack of a centralized entity that controls every cell of an organism and the fact that communication between cells is mostly restricted to their immediate locality, systems in biology and medicine rely on emergence in order to construct complex patterns and produce an outcome. Because of the many levels at which feedback loops can take place and the non-linear effects that small changes to the system can have, it is unlikely that biological systems could be modeled using a set of simple equations, similar to the Newtonian laws of physics [123, 223]. Yet, given sufficient data, representation learning could potentially uncover much structure that is hidden away in the layers of emergent complexity. An example where representation learning has recently successfully overcome the limitations of past approaches is the prediction of protein folding [116].

Protein folding is a multi-stage process where the final structure of the protein is non-trivially connected to the mRNA (messenger RNA) sequence of the protein. An mRNA strand encodes amino acids, which form the primary structure of a protein. The charges of individual amino acids and additional proteins such as chaperones then guide the folding of the protein into the

[1]There is still some engineering involved to align inductive biases of models with the problem one is trying to tackle. We will touch on this later.

secondary and tertiary structures. Finally, tertiary structures combine to form the quaternary structure which is typically the active form of the protein. The function of the protein is then defined by its 3D structure and thus emergent from the folding process given the sequence [2, 61]. As there are more potential folding configurations than atoms in the universe the search space of structures is generally intractable, such that the problem of protein folding has remained unsolved for a long time. Only recent Deep Learning approaches have been able to make some progress on this front by learning heuristics from large amounts of data [116].

Thesis Structure and Contributions This thesis begins with an introductory chapter that introduces some basic concepts from representation learning and algebraic topology that are used to develop methods in this thesis. The chapter concludes with a summary of the potential that representation learning can have in biology and healthcare and a section that describes the contributions I had to the works presented in this thesis that were published under joint authorship. After the introduction, I present the main contributions of the thesis. They are split into three parts:

* * *

In part I – Dimensionality Reduction, I present a novel topology-inspired approach to non-linear dimensionality reduction that leverages the power of autoencoders to derive low-dimensional representations of input data. Dimensionality reduction is an important method used for hypothesis generation in the life sciences, yet few methods are suited to handle the multi-scale relationships present in biological data. The presented approach has the unique capability of capturing similarities of the input data at multiple scales and shows excellent performance on a synthetic dataset as well as a series of image datasets.

* * *

In part II – irregularly-sampled sequences, I present methods and studies related to irregularly-sampled data in the medical domain. More precisely, in Chapter 3 I present a new approach for the classification of MALDI-TOF MS spectra. MALDI-TOF MS data is of high importance in the clinic and is used to analyze the composition of samples and to determine the presence of microbes and their resistances. Only a few machine learning approaches have been specifically developed for this data and all rely on binning of the spectrum which leads to information loss. In contrast, I present an approach that can operate directly on the peaks of MALDI-TOF MS spectra by constructing a kernel that operates on said peaks. I show that this leads to significantly higher performance on an antibiotic resistance prediction task and allows the rejection of out-of-distribution samples due to good uncertainty quantification when combined with a Gaussian process classifier.

Further, in Section 4.2 I present an alternative approach for classifying irregularly-sampled time series data. Irregularly-sampled medical time series are omnipresent in electronic health records and thus of great importance for machine learning in healthcare. Unfortunately, most machine learning algorithms are not compatible with irregularly-sampled data leading to information loss due to the necessity of binning or imputing values. The approach I propose is based on rephrasing the classification of irregularly-sampled time series as a set classification problem. I demonstrate the viability of this approach on medical time series data, where it can be uniquely applied without prior binning or imputation.

Moreover, in Section 4.3 I present an evaluation of medical time series models with a focus on their capability to transfer to other datasets. As treatment protocols vary largely between locations, the transferability of models trained on data from one region to data from another region is highly questionable. The study presents the first harmonized multi-region dataset with a focus on sepsis early prediction. It highlights that the derivation of appropriate labels and the avoidance of circularity are critical for the successful application of machine learning models to clinical data. Further, it shows that in contrast to common expectations, deep learning models seem to be *less* sensitive to distribution shifts between hospitals.

* * *

In part III – Graphs, I present work on improving the performance of graph neural networks and evaluating the predictions of graph generative models. Chapter 5, presents a graph neural model for improved supervised learning on graphs. Improving the expressivity and performance of graph neural networks is an important task, as many GNNs are upper bounded by the Weisfeiler-Leman test. The presented approach improves upon this expressivity bound by allowing the model to access multi-scale information exposed through the framework of persistent homology. Importantly, it does so without requiring the pre-computation of additional graph features or by requiring a significantly higher computation budget. It shows higher classification performance especially on many graph learning tasks, especially when node labels are not present.

In Chapter 6, I show the importance of correctly evaluating graph generative models. Graph generative models represent a relevant methodology for pharmaceutical applications where they could be used to generate small molecules with desirable properties. The quantitative evaluation of graph generative models is crucial as we humans are not able to intuitively reason about the similarity of graphs. Most models are evaluated using maximum mean discrepancy combined with a graph descriptor function. The study shows that the correct selection of hyperparameters is essential for correctly estimating performance and that incorrect selection can lead to arbitrary rankings. Further, I present a sensitivity analysis of different hyperparameter selections using graph perturbations.

* * *

Finally, I conclude the thesis in Chapter 7, where I present a holistic summary of the contributions of the thesis and provide an outlook over future research directions. The following section begins the introduction by describing the importance of deriving the right representation in the context of machine learning.

1.1 The Importance of Representations for Machine Learning

The importance of the right representation has been known in machine learning and statistics for decades. Multiplying numbers is a classic example of how important the right representation is. If we were given numbers to multiply in Roman numerals, the first thing we would do is convert them to Arabic digits so we can apply the algorithms of multiplication we learned in high school. While the information content between the Roman and Arabic numerals is the same, we can work more efficiently with the Arabic representations (be it due to the fact that we learned how to use it or because a system where each position represents a digit can more easily handle carry-over operations) [87].

In machine learning the representation of the input data is hypothesized to be important as it determines which aspects of the data are highlighted, entangled, and potentially also hidden. A perfect representation would thus only encode the explanatory factors in the data that are of relevance for the task at hand and thus form abstractions [18]. This process of removing unneeded information for the task at hand is hypothesized to also be present in our brains [172].

In general, there have been two approaches to construct representations in machine learning: feature engineering and feature learning (also called representation learning or deep learning). We will discuss these in the following subsections while putting more focus on representation learning as it is the main focus of this thesis.

1.1.1 Engineering Representations

Much applied machine learning research has focused on *feature engineering*, where human knowledge is exploited to construct the representation used to fit a statistical model. This allows the human to decide through which "perspective" a problem should be viewed. For example, while the XOR-operation $y = b_1 \text{ XOR } b_2$ between two binary numbers b_1 and b_2 cannot be learned using a linear classifier, a linear classifier can correctly classify the outcome using a transformed representation of the input data where $b_1' = b_1 b_2$. Typical machine learning models which rely on a feature engineering step prior to their application are "low-capacity" models (i.e. models with limited flexibility in determining the decision surface) such as linear/logistic regression, decision trees, and nearest neighbor models.

It is important to note that linear classifiers can also classify non-linear data if provided with the right representation (as shown in the XOR example above) yet cannot derive the non-linear representation themselves as all components being optimized on the data are linear. One approach to constructing non-linear classifiers is using kernel methods, where the representation is *implicitly* defined via a kernel function, such that the effective feature space can even be infinite-dimensional. These rely mostly on local smoothness properties in order to derive predictions, such that predictions are a (potentially non-linear) weighted combination of the training labels of instances that are considered close from the perspective of the kernel function. Interestingly, some parametric kernels can also be learned via optimization such that kernels can be seen as at the intersection of feature engineering and representation learning. For more details on kernels and their usage in the context of representation learning, I refer interested readers to Section 3.1.

1.1.2 Learning Representations

In machine learning, representation learning (or also *feature learning*) describes approaches that aim at deriving representations of raw input data that make the extraction of information easier *with limited human engineering*. This is often with the goal of solving a single task or set of specific downstream tasks [18]. Many current approaches derive representations from a long series of non-linear transformations, where each non-linear transformation is summarized as a “layer”. As many layers are chained creating “deep” stacks of transformations, this practice is often referred to as *deep learning* [87]. The simplest type of neural network used in deep learning is the fully connected feed-forward neural network, which we will formally introduce below.

Artificial Neural Networks (ANNs)

Artificial neural networks (ANNs) are a class of function approximators that take inspiration from biological neural networks and use similar language to describe them. Nevertheless, artificial neural networks are typically much more simple than their biological counterparts as much of the neuron processing is modeled by linear signal aggregation and a non-linear activation function which determines if a neutral is “active” and thus would propagate signals to neurons connected to it. The simplest Artificial Neural Network is the fully connected feedforward neural network or multilayer perceptron (MLP) which we will briefly introduce below.

Fully connected feedforward neural networks are composed of stacks of affine transformations separated with non-linearities.

Definition 1.1 (MLP)**.** A fully connected feedforward neural network g of depth n is a stack of n affine transformations $f_i(x) = A_i x + b_i, i = 1..n$ separated by non-linearities $\sigma_i, i = 1..n$

$$g(x) = \sigma_n \circ f_n \circ \sigma_{n-1} \circ f_{n-1} \circ \cdots \circ \sigma_1 \circ f_1(x) \tag{1.1}$$

where $f \circ g$ denotes the composition of functions f and g.

Without interspaced non-linearities, the stack of affine transformations would be mathematically equivalent to a single affine transformation and thus not more expressive than a single affine transformation. Networks with many layers have higher depth and thus can be considered *deep* neural networks.

A common non-linearity used in deep learning is the rectified linear unit (ReLU), which is defined as $\sigma(x) = \max\{0, x\}$. The ReLU preserves many properties which make linear models easy to optimize via gradient descent. An "active" ReLU with $x > 0$ passes gradients through unchanged and thus does not lead to vanishing gradients as was the case with sigmoid non-linearities used in early iterations of neural networks in the 1980s. Nevertheless, the unit does not pass any gradient when it is inactive which can also lead to problems of "dead" ReLUs that remain inactive during the complete course of training [145].

Gradient-Based Optimization of Artificial Neural Networks Using Back-Propagation

Modern neural networks are almost exclusively trained using variants of stochastic gradient descent with gradients derived by applying the back-propagation algorithm [87]. Stochastic gradient descent optimizes the parameters of the model following the direction of improvement (i.e. the negative gradient with respect to the error signal) which is estimated on subsets of the data.

Definition 1.2 (Stochastic gradient descent)**.** Stochastic gradient descent is an optimization method that optimizes the parameters θ of a model g_θ iteratively using gradients estimated on subsets of the data $\nabla l(g_\theta(\tilde{x}), \tilde{y}), \tilde{x}, \tilde{y} \subset \mathcal{D}$, where l represents a loss function. The parameters are updated according to a learning rate η in the opposite direction of the gradient:

$$\theta_{t+1} = \theta_t - \eta \nabla l(g_{\theta_t}(\tilde{x}), \tilde{y}) \tag{1.2}$$

While in many cases the gradient can be explicitly derived, the evaluation of the derivation can often result in numerical issues/instability and is inefficient [87]. Most frameworks instead rely on the back-propagation algorithm in order to derive the parameter gradient from the error signal. In back-propagation, a forward pass or forward propagation is performed, where the data is passed through the network to produce an output and a loss. This error signal is then converted into gradients by essentially following the chain rule for derivatives from the loss backward, which is referred to as the backward pass. By reusing computations from layers closer to the loss, back-propagation is more efficient and avoids instabilities due to numerical precision by ensuring that gradients are consistent over the backward pass.

To clarify things, consider a simple example of a two-layer MLP with $g(x) = W_2\sigma(W_1x + b_1) + b_2$. We can simplify the notation by summarizing the layers as functions with parameters θ, s.th. $g(x) = f_2(f_1(x))$ and denote intermediate representations between layers as $h_1 = f_1(x)$ and $h_2 = f_2(h_1)$. Using the chain rule one can compute the gradient of an input with respect to the loss $l(g(x), y)$ as

$$\frac{dl}{dx} = \frac{dh_1}{dx}^\top \frac{dh_2}{dh_1}^\top \frac{dl}{dh_2}^\top \tag{1.3}$$

where $\frac{dh_1}{dx}$ is the Jacobian matrix of f_1, $\frac{dh_2}{dh_1}$ is the Jacobian matrix of f_2 and $\frac{dl}{dh_2}$ is the Jacobian matrix of $f_3(x) = l(\cdot, y)$. The error signal at layer i is often denoted as $\delta_i = \frac{dh_{i+1}}{dh_i}^\top \frac{dh_{i+2}}{dh_{i+1}}^\top \dots \frac{dl}{dh_d}^\top$ and can thus be computed in a recursive fashion starting from the output of the network. The gradients for the parameters of each layer are then computed with respect to the error signal applied to the layer. For example, in the case of transformation f_1 it holds that

$$\frac{dl}{dW} = W_1^\top \sigma' \delta_1 \tag{1.4}$$

such that the gradient computation can efficiently reuse past computation steps. Back-propagation essentially represents the recursive application of the chain rule, to decompose the gradient computation into reusable components.

Inductive Biases for Learnt Representations

While MLPs have universal approximation guarantees, i.e. are able to approximate arbitrary functions to arbitrary precision if constructed to be large enough, some functions can be more easily represented than others. The preference of or constraint to certain functional classes is referred to as *inductive biases* [15, 38] and is determined by the model architecture. Many such architectures exist and have been developed over the years a subset of which I will introduce in the following paragraphs while concentrating on the types of data they support and their inductive biases. For a more mathematical description, I refer the interested reader to Goodfellow, Bengio, and Courville [87] for an explanation of most architectures developed prior to 2016 and Vaswani et al. [239] for the more recently developed transformer architecture.

Convolutional Neural Networks Convolutional Neural Networks (CNNs) have been one of the main drivers of Deep Learning's success in the image, video, and sound domains. A

CNN layer is composed of a set of filters that are applied to the input data via convolution, similar to filters in image processing. This results in a transformed representation of the input image (approximately the same size as the image) where different filters focus on different patterns present in the data. Similar to MLPs a non-linearity (typically ReLU) is applied to the transformed representation before the output is passed on to the next layer.

Typically, CNN architectures increase the number of filters with increasing depth, while reducing spatial resolution such that the number of features stays approximately the same. Spatial resolution is decreased by applying pooling layers that compute the average (average pooling) or maximum (max pooling) over a local set of spatial positions. This is necessary, as the convolution operation applied in CNNs is local and thus would require very deep architectures to propagate information from spatially distant locations. Further, pooling reduces the number of operations that need to be performed and thus also increases the scalability of architectures [117].

Convolutional neural networks are the dominant architecture when working with data that show regularly structured local relations among features, such that patterns can be detected on a local scale. Essentially, CNNs perform a variant of template matching, not only on the raw input image but also on transformed variants thereof. In the example of an image the first few layers typically detect edges of varying orientation, the middle layers recognize smaller structures (such as cat nose, ears, and mouth) and later layers integrate the outputs of middle layers to a potentially abstract concept (i.e. a cat)[2]. Similar notions of an increasingly coarse structure are present in video and audio data, which explains why CNNs have been successful in these domains [13, 177].

A further reason for the effectiveness of CNNs is their inductive bias. Inductive bias describes the set of assumptions a model relies on to solve the learning task at hand. CNNs represent translation equivariant functional mappings, such that the absolute position of an input is not relevant to the output of the function, and if the input is shifted, the output is also shifted accordingly. In the task of image classification, a cat remains a cat independent of it being in the bottom right or top left of the image, such that translation equivariance is a very influential property for a model to perform well in this task.

Recurrent Neural Networks Recurrent neural networks (RNNs) are networks that propagate state information over a sequence of inputs. At each step (typically time is the axis along which RNNs are unrolled), the model processes the input conditional on its internal, updates the internal state, and outputs a value. As an RNN outputs a value at each step, RNNs can also be stacked in layers to form multiple levels of processing in order to increase expressiveness. RNNs

[2]This example, of course, assumes that the CNN was trained to recognize cats, and provides an intuitive grasp. In practice, it is hard to exactly pinpoint the responsibility of filters, but some progress has been made in this direction [43].

are typically used to process sequences of varying lengths, although their dominance in this realm has been questioned in recent years [13].

While RNNs have been developed for a long time, they only became successful with the development of the Long Short-Term Memory (LSTM) RNN [96], which addressed a fundamental issue in RNN training namely, exploding and vanishing gradients. This allowed the research community to tackle more challenging sequential tasks like speech recognition. Vanilla RNNs are very difficult to train as they essentially represent a very deep neural network where depth in this case can be measured as the length of the input sequence. Let us denote the function f as the mapping from one state of the RNN to another. Due to the repeated application of f, the scale of outputs will strongly depend on the behavior of f, more precisely on the eigenvalues of its Jacobian. As shown in Section 1.1.2 the computation of gradients is essentially a series of multiplications of Jacobian matrices of transformations involved in computing the loss, and thus for RNNs corresponds to a matrix power of the Jacobian matrix of f. If the largest eigenvalue of the Jacobian matrix of f is larger than 1 the output of these computations will grow exponentially and thus the computed gradients will explode. Further, if the largest eigenvalue of the Jacobian is less than 1 gradients will shrink exponentially fast to zero [181]. Hochreiter and Schmidhuber [96] suggest a mechanism, which allows parts of the gradient to be passed through without undergoing any transformations, which ensures the largest eigenvalue of the Jacobian matrix is 1 for many time steps and thus significantly improves training stability.

Topology as a Source for Inductive Biases In this thesis, I present many methods that draw inspiration from the field of topology and incorporate topology-specific computations into the regular computations of neural networks. This allows computations of these augmented neural networks to directly leverage topological invariants and information from multiple scales of the data. As these augmentations require some background in the methods from topology they take inspiration from, the following section will provide a brief introduction to the concepts from topology that are of relevance to this thesis.

1.2 Representation Learning With Topological Inductive Biases

Many of the methods presented in this thesis rely on perspectives, notions, and formalisms from the field of topology. This warrants a dedicated introduction to the subfields of topology that are of relevance to this thesis and can be found in the following sections.

Topology is the field of mathematics that studies the properties and invariants of topological spaces, in particular under specific types of transformations — so-called homeomorphisms. For the means of this thesis, a topological space is any set of points with a defined neighborhood relationship. In contrast to, for example, the space in which we physically reside — the (Euclidean) metric space — this neighborhood relationship does not require any notion of magnitude and also does not have to follow any specific properties such as the triangle inequality. It is thus an inherently discrete notion of a space that solely captures which points or positions can be considered neighboring.

Homeomorphisms on topological spaces define spaces that should be considered "equivalent" from a topological standpoint. For example, a point should still be considered in the neighborhood of another point even when their distances are scaled, all points are rotated or all points are shifted to a new location. Two spaces are considered homeomorphic if there exists a mapping (i.e. a homeomorphism) which allows us to smoothly move between these spaces while maintaining neighborhood relationships. More formally, a homeomorphism is a bijective continuous mapping $f : \mathbb{X} \to \mathbb{Y}$ between two topological spaces $\mathbb{X}$ and $\mathbb{Y}$ where the inverse mapping $f^{-1} : \mathbb{Y} \to \mathbb{X}$ is also continuous. Further, if f is bijective then f^{-1} must also naturally be bijective, the additional continuity constraints allow us to smoothly transition between the two spaces. It is important to note that in this case, continuity refers to the topological notion of continuity which requires that the preimage of any open set is also open. Interestingly, this coincides with our natural understanding of continuous functions on real-valued spaces.

1.2.1 Algebraic Topology

Algebraic topology provides tools to study topological spaces by associating algebraic structures such as groups and vector spaces to them that remain invariant under homeomorphisms [93]. In the composition of this section I relied on various previous works such that notations and analogies might overlap with the following publications:

- B. Rieck. "Persistent Homology in Multivariate Data Visualization". PhD thesis. Ruprecht-Karls-Universität Heidelberg, 2017
- F. Hensel, M. Moor, and B. Rieck. "A Survey of Topological Machine Learning Methods". *Frontiers in Artificial Intelligence* 4, 2021
- M. Horn* et al. "Topological Graph Neural Networks". In: *International Conference on Learning Representations*. 2022

Simplicial Complex One of the most important structures in algebraic topology is the notion of a simplicial complex. A simplicial complex K is a structure that can be seen as a high-dimensional generalization of a graph. It is composed of simplices of different dimensions, such as

0-dimensional simplices (representing vertices) and 1-dimensional simplices (representing edges). Besides representing structural elements of graphs, simplices can also represent higher-dimensional structures such as triangles, which are then considered 2-dimensional. Each simplex $\sigma \in$ K has a potentially empty set of faces where each face τ needs to necessarily be part of the simplicial complex $\tau \in$ K. Further, non-empty intersections of simplices should also be part of the simplicial complex i.e. if $\sigma \cap \sigma' \neq \emptyset$ for $\sigma, \sigma' \in$ K then $\sigma \cap \sigma' \in$ K. More informally, this means that a simplicial complex is closed under the computation of faces of simplices.

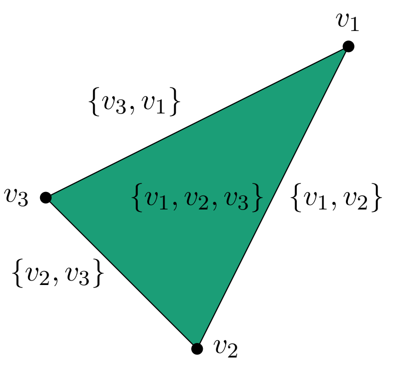


Figure 1.1: Illustrative example of a simplicial complex K modelling a triangle.

As an illustrative example, let us consider the simplicial complex K of a triangle. The triangle has vertices corresponding to the corners of the triangle $\{\{v_1\}, \{v_2\}, \{v_3\}\}$, and edges that connect the nodes together $\{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_1\}\}$. Finally, we include the face of the triangle $\{\{v_1, v_2, v_3\}\}$ to highlight the possibility of higher dimensional structures than those conventionally found in a graph (which is also considered a 3-clique). The complete simplicial complex is the union of all these structures and obeys the rules defined above, i.e. K $= \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_1\}, \{v_1, v_2, v_3\}\}$. The intersections of the edges are the individual vertices which are part of the simplicial complex itself, further vertices have empty faces and the faces of the edges are defined by the vertices they connect, which are also part of the simplicial complex.

Chain groups With simplicial complexes as the basic building blocks, we can define further notions in terms of vector spaces and groups to obtain computable representations. In particular, the d^{th} chain group of a simplicial complex K, is denoted as $C_d(K)$ and represents the vector space generated over all formal linear combinations of d–dimensional simplices in K such that all elements of C_d are of the form $\sum_j m_i \sigma_j$ for $\sigma_j \in$ K and $m_j \in \mathbb{Z}_2$. $\mathbb{Z}_2$ is the finite field of two elements (i.e. $\{0, 1\}$) and thus amounts to either including or excluding individual simplices. The chain group over $\mathbb{Z}_2$ can thus also be thought of as all finite unions of simplices of a certain dimension. While it is not generally required to compute chain groups solely over $\mathbb{Z}_2$, this will be the only case that we consider in this thesis.

The elements of $C_d(K)$ are referred to as *simplicial chains*. For $C_d(K)$ to represent a group, we need to additionally define the group operation on two simplicial chains. For two simplicial chains a and b the group operation $+$, or more precisely the addition with $\mathbb{Z}_2$ coefficients, is defined as

$$a + b = (a \cup b) \setminus (a \cap b)$$

It is thus equivalent to the symmetric difference between the two simplicial chains. This formalism allows us to now perform algebraic operations on representations derived from topological spaces and allows us to better reason about notions such as boundaries and holes of spaces. Moreover, this operation is also easy to implement, making $\mathbb{Z}_2$ the preferred coefficient field for algebraic topology.

Boundary Homomorphism Given a simplicial complex K, the d^{th} boundary homomorphism ∂_d is a linear function between the d^{th} chain group and the $(d-1)^{\text{th}}$ chain group $\partial_d \colon C_d(\mathrm{K}) \to C_{d-1}(\mathrm{K})$ that assigns each simplex to its boundary. The boundary of each simplex or simplicial chain itself is thus again a simplicial chain. For a p-simplex $\sigma = \{v_1, \ldots, v_{d+1}\}$ it is defined as

$$\partial_d \sigma := \sum_{i=1}^{d+1} \{v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{d+1}\},$$

i.e. each element v_i is left out of the simplex once. The calculation extends to the computations on chain groups due to the linearity of the boundary operator.

Looking back to our previous triangle example, we can show how the computation for this specific case would take place. In order to compute ∂_2 of our triangle K we collect the simplicial chains of dimension 2 form K and apply the boundary operator: $\partial_2\{v_1, v_2, v_3\} = \{v_1, v_2\} + \{v_2, v_3\} + \{v_3, v_1\}$. This expression *cannot* be further reduced as it is required to be an element of the chain group C_{d-1}. The boundary of the triangle face is thus the formal sum of its edges.

It is important to note that the kernel and image of the boundary operator are well-defined. In particular, the kernel $\ker \partial_d$ contains all simplices that do not have any boundary, these are typically referred to as d-cycles. Further, the image of the boundary operator $\operatorname{im} \partial_d$ contains the boundaries of all d simplices.

1.2.2 Simplicial Homology

One of the main characteristics of a topological space is the "number of high-dimensional holes" it contains. The study of spaces based on these characteristics is called homology. In order to formally derive a definition of what a high-dimensional hole actually represents, we need to understand how the presence of holes connects to our previous formalism of simplicial complexes, chain complexes, and the boundary operator.

Homology groups To build intuition, let us consider our triangle example once again. The simplicial complex K we considered earlier does not contain any holes, as we consider the triangle $\{v_1, v_2, v_3\}$ to be part of the simplicial complex. Yet if we remove the triangle from our simplicial complex, a hole will be present. This is due to the fact that the boundary of the triangle contains exactly the elements that in the lower-dimensional simplex formed the hole we observed. This is captured formally in the notion of homology groups.

Definition 1.3 (Homology Group)**.** The dth homology group of a simplicial complex K is defined as

$$H_d(K) := \ker \partial_d / \operatorname{im} \partial_{d+1}$$

where the /-operator refers to the quotient group, i.e. the exclusion of elements. Following the intuition we built, cycles should only be considered "real cycles" if they do not represent the boundary of any higher dimensional simplex. This group has a rich structure in that it contains whole classes of cycles, which can be considered equivalent.

Nevertheless, what does it mean that cycles would be considered equivalent in this context? As shown in Figure 1.2, we can draw an arbitrary number of 1-cycles on a sphere, yet from a topological standpoint, all of these would be considered equivalent as they could be formed into each other without tearing the sphere surface. Moreover, strictly speaking, these are not even true cycles! They are all part of the boundary defined by the surface of the sphere, which represents a 2-cycle or void. Thus despite us being able to draw arbitrarily many of these cycles, there is actually not even a single true cycle present and the number of independent 1-cycles would be considered 0. The surface of the sphere, on the other hand, represents a 2-cycle that could be arbitrarily rotated or deformed and still considered part of the same cycle class as all cycles would be enclosing the same void (i.e. the inside of the sphere) and thus capture the same property of the topological space. As previously illustrated, this concept allows us to reason about cycles in higher

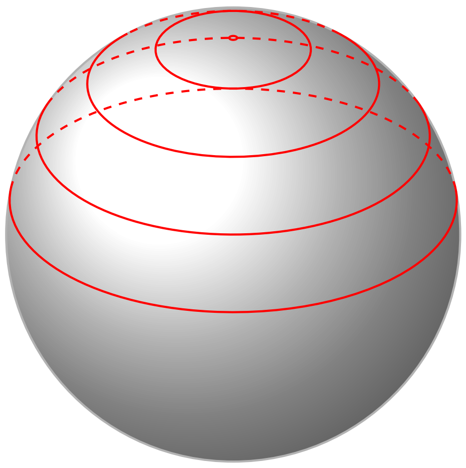
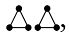
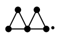


Figure 1.2: Example of sphere containing zero 1-cycles and a single 2-cycle.

dimensional structures and further allows us to use computations to detect or count the number of holes in a topological space. While it is possible to transfer these notions to structures of arbitrary dimension, this thesis focuses on 0 and 1 dimensional homology, which is computationally very tractable [195].

For a more straightforward lower dimensional example, we can also turn our attention back to triangles. Given two triangles , we can recognize the presence of two cycles. These same cycles continue to exist even when the two triangles are connected to form . Thus, while the representation of the cycles or their exact manifestation changed, they would still be considered to represent the same homology class. It is important to note, though, that while in this example the number of 1-cycles remains the same, the number of connected components does change when the two triangles are fused. Thus both representations would not be considered equivalent from the perspective of the 0th homology group.

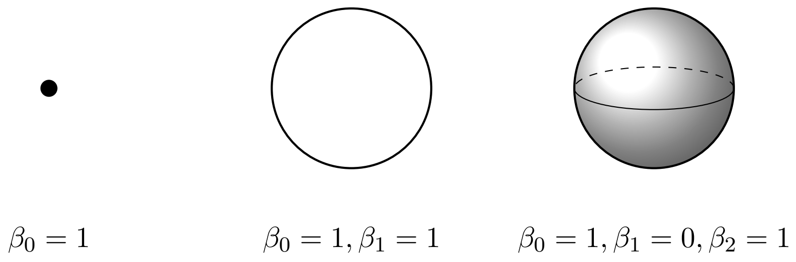

$\beta_0 = 1$
$\beta_0 = 1, \beta_1 = 1$
$\beta_0 = 1, \beta_1 = 0, \beta_2 = 1$

Figure 1.3: Examples of topological structures and their associated Betti numbers. From left to right: point, circle, sphere

Betti numbers The most simple and straightforward summary of a homology group are the so-called Betti numbers. In essence, they quantify the number of non-equivalent homology classes present in the homology group. While this leads to a lot of structural information in the group being lost, the number of classes is good at distinguishing simple spaces. It can be obtained by the group-theoretic notion of rank, which characterizes the minimal number of elements required in the *generating set* in order to give rise to the full group.

Definition 1.4 (Betti numbers)**.** Given a homology group $H_d(\mathrm{K})$ its Betti number is defined as

$$\beta_d := \operatorname{rank} H_d(\mathrm{K})$$

Thus Betti numbers represent counts of distinct cycles that would be necessary to capture the structure of the space. Less formally, we can refer to Betti numbers as counts of d-dimensional holes. Some examples of topological structures and associated Betti numbers are provided in Figure 1.3.

1.2.3 Persistent Homology

The notion of homology discussed above is inherently discrete and represents a very coarse description of the underlying topological spaces. Persistent homology extends these notions to allow more nuanced reasoning over the structure of topological spaces. The core concept of persistent homology is to compute homology not only on a single version of a simplicial complex but on a nested sequence of simplicial complexes — a *filtration*.

Definition 1.5 (Filtration)**.** The filtration of a simplicial complex K is a nested sequence of simplicial complexes $\mathrm{K}^{(0)} \subseteq \cdots \subseteq \mathrm{K}^{(m)}$ such that

$$\emptyset = \mathrm{K}^{(0)} \subseteq \mathrm{K}^{(1)} \subseteq \ldots \mathrm{K}^{(m-1)} \subseteq \mathrm{K}^{(m)} = \mathrm{K}.$$

More precisely, this is a sublevel set filtration; however, as this is the only type of filtration of relevance for this thesis, it will be used to define the general concept. A filtration is typically implemented using a filtration function $f\colon \mathrm{K} \to \mathbb{R}$, such that $K^{(i)} := \left\{\sigma \in \mathrm{K} \mid f(\sigma) \leq f^{(i)}\right\}$. Each subset of a sublevel set filtration only contains the simplices whose function value is less than or equal to the threshold. To ensure that all simplicial complexes in the filtration are valid, the filtration function needs to satisfy the following requirement: The filtration values of faces of simplices need to necessarily be lower or equal to the filtration value of the simplex itself, i.e. $\max_{\tau \in \sigma} f(\tau) \leq f(\sigma)$. In general, we can implement this by requiring the filtration to be *monotonic* in the simplices, i.e. $f(\tau) \leq f(\sigma)$ for all $\tau \subseteq \sigma$.

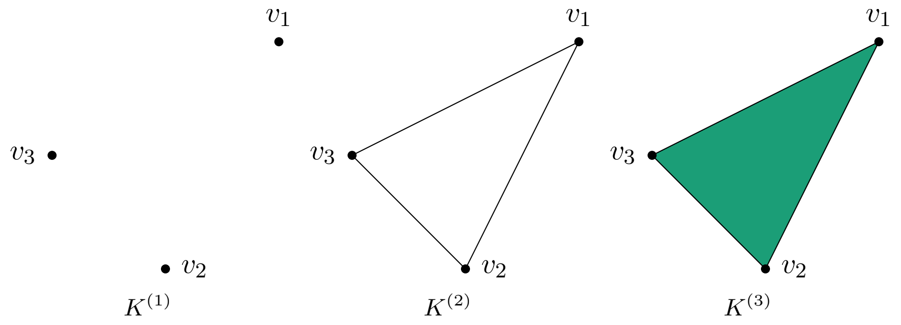


Figure 1.4: Example filtration applied to the triangle example.

In contrast to simplicial homology, this approach can potentially extract more information from the topological space as it allows tracking changes in topological features. For example, if we assume the following filtration function $f : \mathrm{K} \to \mathbb{R}$ for our triangle example

$$f(\sigma) = \begin{cases} c_0 & \text{if } \sigma = \emptyset \\ c_1 & \text{if } \sigma \in \{v_1, v_2, v_3\} \\ c_2 & \text{if } \sigma \in \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_1\}\} \\ c_3 & \text{if } \sigma \in \{\{v_1, v_2, v_3\}\} \end{cases}$$

with $c_0 < c_1 < c_2 < c_3$ we can observe an evolution of the simplicial complex as visualized in Figure 1.4. We observe the creation of the vertices, i.e. 3 connected components such that $\operatorname{rank} H_0(\mathrm{K}^{(1)}) = 3$, whereas at the next step in the filtration two connected components are destroyed and a cycle is created such that $\operatorname{rank} H_0(\mathrm{K}^{(2)}) = 1$ and $\operatorname{rank} H_1(\mathrm{K}^{(2)}) = 1$. Finally, the cycle is destroyed and the triangle is created: $\operatorname{rank} H_1(\mathrm{K}^{(3)}) = 0$ and $\operatorname{rank} H_2(\mathrm{K}^{(3)}) = 1$.

Persistent homology tracks these changes by associating each topological feature, i.e. a connected component, cycle, or void, with a creation and destruction value, corresponding to the filtration value at which simplices get added and removed from the homology groups. These values are recorded as tuples of the form $(f^{(i)}, f^{(j)}) \in \mathbb{R}^2$. It is important to note that some features never get destroyed during the course of a filtration. For example, one connected component continues to exist in the triangle and the triangle face itself cannot be destroyed as we have no higher-order simplices in our simplicial complex to destroy it *with*. It is common practice to associate topological features that are never destroyed with a tuple of the form $(f^{(i)}, \infty)$ and thus consider its destruction at a filtration value of infinity. While there are ways to avoid this using the notion of *extended persistence* these are computationally considerably more demanding [53].

Persistent homology groups The question that remains in the above example is how to associate the topological features between different steps of the filtration. Due to the requirements for the construction of a filtration, there is a clear connection between the different steps of a filtration. Formally, the steps are connected by the inclusion homomorphism between $\mathrm{K}^{(i)} \subseteq \mathrm{K}^{(i+1)}$. The composition of the inclusion homomorphism with the boundary homomorphism itself induces a homomorphism between the corresponding homology groups of the filtration. We will denote this map with $\iota_d^{(i,j)} : H_d(\mathrm{K}^{(i)}) \to H_d(\mathrm{K}^{(j)})$, it thus maps from the homology group of the filtration at level i to the homology group of the filtration at level j and gives rise to a sequence of homology groups

$$H_d(\mathrm{K}^{(0)}) \xrightarrow{\iota_d^{(0,1)}} H_d(\mathrm{K}^{(1)}) \xrightarrow{\iota_d^{(1,2)}} \dots \xrightarrow{\iota_d^{(m-1,m)}} H_d(\mathrm{K}^{(m)}) = H_d(\mathrm{K})$$

for each dimension d. We can further denote the mapping between arbitrary subsequent elements of the filtration as a composition of the individual maps such that $\iota_d^{(i,j)} = \iota_d^{(j-1,j)} \circ \cdots \circ \iota_d^{(i+1,i+2)} \circ \iota_d^{(i,i+1)}$ for all $i \leq j$, where we simply compose all consecutive mappings between from the filtration at level i to the filtration at level j. For example, the mapping from level 1 to level 3 would be defined as $\iota^{(1,3)} = \iota^{(2,3)} \circ \iota^{(1,2)}$.

With this formalism, we can extend to notion of homology groups to persistent homology groups. The d^{th} persistent homology group of a simplicial complex K with a compatible filtration f is defined as

$$H_d^{(i,j)} := \ker \partial_d\left(\mathrm{K}^{(i)}\right) / \left(\operatorname{im} \partial_{d+1}\left(\mathrm{K}^{(j)}\right) \cap \ker \partial_d\left(\mathrm{K}^{(i)}\right)\right)$$

for $i \leq j$. Thus the persistent homology group is a generalization of the homology group to different levels of a filtration and essentially contains all elements of $H_d(\mathrm{K}^{(i)})$ that are still present in $H_d(\mathrm{K}^{(j)})$. The intersection with $\ker \partial_d\left(\mathrm{K}^{(i)}\right)$ is solely necessary to guarantee that we would only remove elements that are actually contained in $\ker \partial_d\left(\mathrm{K}^{(i)}\right)$.

Taking a step back, we can transfer the notion of Betti numbers to persistent homology groups by defining them in terms of the group rank, i.e. $\beta_d^{(i,j)} := \operatorname{rank} H_d^{(i,j)}$. As a strong simplification, we can say that persistent homology generates a sequence of Betti numbers instead of only a single number for each dimension. Nevertheless, we can extend this notion to additionally understand which individual homology classes are created at which step in the filtration and when they become a part of a higher level feature, which will give us a more nuanced picture.

In particular, we can say that a class $c \in H_d(\mathrm{K}^{(i)})$ is created in $\mathrm{K}^{(i)}$ if it is not an element of an earlier persistent homology group, i.e. $c \notin H_d^{i-1,i}$. This is due to the nesting relationship induced by the filtration. If c were created in an even earlier filtration step $H_d^{i-2,i-1}$ then it would remain an element of the persistent homology group except if it were destroyed by the formation of some other class. Yet if this were the case, then c would not be an element of $H_d(\mathrm{K}^{(i)})$ forming a contradiction. Further, we can say the same class $c \notin H_d(\mathrm{K}^{(j)})$ is destroyed in $\mathrm{K}^{(j)}$ if the class is merged into another class while traversing the filtration from $\mathrm{K}^{(j-1)}$ to $\mathrm{K}^{(j)}$. Formally, $i_d^{(i,j-1)}(c) \notin H_d^{i-1,j-1}$, i.e. the homology class corresponding to c after applying the filtration steps from i to $j-1$ is not part of any earlier persistent homology group $H_d^{i-1,j-1}$ (which represents those homology classes of $\mathrm{K}^{(i-1)}$ that are still present in $\mathrm{K}^{(j-1)}$) and $i_d^{(i,j)}(c) \in H_d^{i-1,j}$, i.e. that the homology class of the merged feature $i_d^{(i,j)}(c)$ is part of the persistent homology group $H_d^{i-1,j}$. In this case we would say that the class c persists from $\mathrm{K}^{(i)}$ to $\mathrm{K}^{(j)}$.

Persistence diagrams A common summary statistic for each topological feature is its persistence, which defines how long a feature persists.

Definition 1.6 (Persistence)**.** For a homology class c filtered according to a filtration function f, created in $K^{(i)}$ and destroyed in $K^{(j)}$, we define the persistence as

$$\operatorname{pers} c := f^{(j)} - f^{(i)},$$

where the persistence is non-negative pers $\in \mathbb{R}^{+}$[3] due to properties of the filtration function which prevent classes from being destroyed before they are created.

A persistence diagram is a way to visualize the creation and destruction of homology classes during the course of a filtration. In particular, a persistence diagram of dimension d, denoted by $\mathcal{D}_d$, contains the filtration tuples $(f^{(i)}, f^{(j)})$ of all d-dimensional topological features.

1.2.4 Computation of Persistent Homology

To give a more hands-on grasp of what topological features are and how they are derived in practice, I will briefly introduce a simple and fast method for the computation of 0-dimensional persistence. The algorithm is based on the union–find data structure, which stores a disjoint collection of sets and allows them to be queried using the find operation. While this approach is limited to 0-d persistent homology (i.e. the detection of connected components and cycles) it is the algorithm used in this thesis for the derivation of topological features. The union-find-based algorithm is presented in Algorithm 1.

Given the understanding of persistent homology presented in this introduction, it quite straightforward to see where the theory meets the computation. We initialize the union-find data structure – it will be used to keep track of the state of the simplicial complex during the course of the filtration. We then sort the input simplices according to the filtration values and examine if their faces are already present in the union-find data structure. Due to the nesting property of a filtration, this has to be the case for all faces in the simplex. Thus `parents` would only remain empty if the simplex does not have any faces (which for instance can be the case if the simplices represent vertices). We then add the simplices to our simplicial complex by keeping track of them in `U`, but do not register a tuple in the persistence diagram as the creation of 0D objects (i.e. vertices) does not destroy anything.

If the faces of the simplex are already present in `U`, then we derive the representative elements of the corresponding connected components using `U.find`. The representative element is the first element added to the connected component. In the case of 0-dimensional persistence, a simplex can at most have two faces. As the simplex connects its faces, we merge the two connected components that the faces of the simplex are connected to. This is done according to a convention by merging the younger component (i.e. the component which was created later in the filtration pro-

[3] We define $\mathbb{R}^{+} = \{x \in \mathbb{R} \mid x \geq 0\}$

Algorithm 1 Computation of 0-dimensional persistent homology using the union–find data structure.

```
Require: Sequence of simplices $\sigma_1, \dots, \sigma_n$ with associated filtration function $f$
  U $\leftarrow \varnothing$
  $\mathcal{D} \leftarrow \varnothing$
  Sort simplices according to filtration values such that $f(\sigma_1) \leq f(\sigma_2) \leq \cdots \leq f(\sigma_n)$
  for i = 1 to n do
    parents $\leftarrow \varnothing$
    for $\tau \in \sigma_i$ do {Iterate over faces $\tau$ of $\sigma_i$}
      parents $\leftarrow$parents $\cup$ U.find($\sigma_i$) {Get parents of faces}
    end for
    if parents $= \varnothing$ then {Simplex has no parents}
      U.add($\sigma_i$)
    else
      older $\leftarrow$arg min$_{p \in \text{parents}}$ $f(p)$
      younger $\leftarrow$arg max$_{p \in \text{parents}}$ $f(p)$
      if younger $\neq$older then
        U.merge(younger, older) {Merge younger component into older one}
        $\mathcal{D}$.add($f$(younger), $f(\sigma_i)$)
      else {This is a cycle, we do not add a feature to the diagram}
        Keep track of cycles if needed (not part of the 0D algorithm).
      end if
    end if
  end for
  for root $\in$ U.roots do {handle unpaired elements}
    $\mathcal{D}$.add($f(root), \infty$)
  end for
```

cess and thus has a larger filtration value) into the older component (i.e. the component which was created earlier and thus has a lower filtration value), which is reflected by the `U.merge` operation. If both faces already belong to the same connected component, the simplex we are introducing will create a cycle. Cycles are not part of the 0D persistence diagram, such that we do not add any elements to it. We can keep track of cycles as indicated in the algorithm.

After iterating over all simplices, we finally add the connected components that still exist by iterating over `U.roots` and add them to the diagram paired with ∞ as these were not destroyed during the course of the filtration.

1.3 Bridge the Gap to Biological and Medical Applications

Many works in this thesis focus on domains where representation learning algorithms are applied to tackle problems in biology and healthcare. This section will thus briefly discuss why machine learning and in particular representation learning are of great potential for these fields and which topics are being researched at their intersection.

1.3.1 Machine Learning for Biology

While machine learning methods have been utilized in the context of biology for many years [211], the development of high-throughput sequencing [193] and experimentation platforms has made a much wider class of models applicable. There are multiple ways in which machine learning approaches can support biologists in the process of doing research. In the following, I will describe the two main branches to give a broad overview.

The first is to help researchers understand the large quantities of high-dimensional data. The process of understanding data through visualization and the exploration of different representations and transformations is referred to as *exploratory data analysis*. It represents an integral component in hypothesis generation, which can provide new directions for follow-up experiments. Here, machine learning models can support researchers by reducing the complexity of the data via *dimensionality reduction*. In dimensionality reduction, a machine learning model is applied to the data to derive a representation of it that can be visualized and intuitively understood by humans. There exist a large number of dimensionality reduction methods [106, 148, 153, 183, 209, 228] developed for general data visualization and understanding. Recently, there has further been a line of work that concentrates on developing dimensionality reduction methods specifically geared for biology [5, 62, 156, 238]. These approaches incorporate more information specific to the data-generating process and show that further developments in this area have a great potential to accelerate biological research.

The second line I want to highlight is by providing accurate predictions of how biological systems will behave. Here, binding prediction can be used to estimate the potential interactions of a protein with other proteins [33, 120] or DNA/RNA sequences [3, 234]. It can thus be used to generate hypotheses for mechanisms of action or understand regulatory cascades. The prediction of behavior and properties is further relevant in the context of bioengineering and synthetic biology, where the goal is to change or create biological systems and components to solve a defined task. Approaches that accurately predict protein structure from sequences have the potential to aid the development of new biological catalysers [116] and significantly accelerate research in this domain. Further, the prediction of small molecular properties has been of great interest for the pharmaceutical industry to avoid many experiments and focus on those with the largest potential [256]. Recent research has also investigated how machine learning models could be used to predict the behavior of different cell types when exposed to perturbations [112].

Machine learning and representation learning in particular has great potential in shaping how research is done by helping researchers with hypothesis generation and potentially even the simulation of outcomes. Deciphering protein folding has been a very important step towards even more holistic models which might be able to simulate whole cells and model their responses to external stimuli. The work presented in this thesis contributes to the ongoing efforts to understand complex biological systems by suggesting a novel dimensionality reduction approach designed specifically for hierarchies of objects which show similarities at multiple scales (see Chapter 2). Further, I present approaches that can improve the prediction of properties on multi-scale graphs that in turn can help in predicting the functionality of biological systems based on structures such as proteins (using their subdomain connectivity) and small molecules (see Chapter 5). Finally, the evaluation of generated graphs is discussed in this thesis (see Chapter 6), which is relevant for the engineering of new small molecules. These small molecules are of great importance to the pharmaceutical industry.

1.3.2 Machine Learning for Healthcare

The vast amount of biomedical data being collected in modern digital health record systems [113] has made healthcare a domain where machine learning algorithms could provide valuable support to clinicians [154]. The deployment of such systems is already starting to have a measurable impact on physicians, health systems, and patients [233]. The two main areas in which these models have been applied are medical imaging and electronic health records. While the former is not of relevance to this thesis, I will still briefly provide an overview for interested readers.

Due to the success of deep learning approaches on images and the transferability of said architectures to data with similar characteristics, medical imaging has been the area most significantly impacted by recent machine learning advancements. In medical imaging, the goal is to predict health states from an image which can be 2-dimensional (in the case of regular images, X-ray images, and images of probe slides) or 3-dimensional (in the case of computed tomography (CT), positron emission tomography (PET) and magnetic resonance imaging (MRI)). The exact task of the model is often to segment organs or parts of organs that show abnormalities linked to underlying diseases such as cancer. When the object being examined is a slide, then the prediction target can also be disease classification. While insufficient data availability is still an issue in some contexts [269], many approaches have surpassed the performance of practitioners [71] and thus can be assumed to already contribute to better health outcomes. Further, due to the potential reduction of specialist consultations, clinical care in rural areas can be improved by simultaneously reducing costs.

A further line of research lies in the application of machine learning models to electronic health records (EHRs), which encompass information measured by continuous monitoring systems, lab measurements, and clinician notes [190]. These models focus on predicting treatment outcomes, such as mortality (to assess if a patient is at high risk), length of stay (to allow the hospital to better plan the utilization of its available resources), sepsis (to respond to the life threatening systemic infection and provide fast adequate treatment), readmission probability and diagnosis. EHRs represent a highly challenging class of data, as formats from different hospitals are often incompatible and the structure of medical notes can also vary between regions and doctors. As cross-hospital heterogeneity is associated with low generalization performance, much of current research seeks to improve performance in these scenarios [8, 162]. A further issue is the confidentiality of said data where hospitals are often not allowed to share the data with researchers developing models. Here, research is being conducted to apply federated learning methods to machine learning for healthcare [198] which would also models to be trained without the data ever needing to leave the hospital.

Summarizing, over the last few years the number of machine learning methods that are applied in a clinical context has risen significantly and some have already shown great potential to improve the current state of care [233]. While there remain many areas where the gap to clinical application is large, progress is being made at a rapid pace. This thesis concentrates on some of the more technical aspects of how to design models that work with irregularly-sampled time series present in EHRs (see Section 4.2) and irregularly-sampled MALDI-TOF peak data (see Chapter 3). Here different approaches are followed based on the availability of data in the respective domain. Further, we investigate the impact of distribution shifts between hospitals and show that high-capacity models do not necessarily show lower performance in this context (see Section 4.3). I hope the work here can help to improve the state of care in the long term.

1.4 Attributions

During the course of my thesis I had the great pleasure to work together with many highly talented people who show a great passion for the collaborative aspects of science. While this allowed me to work on many different topics in joint efforts together with other Ph.D. students it is also my responsibility to give credit where credit is due. I will thus in the following describe my contributions to the projects in this thesis, when the work was performed jointly with other PhD students. These will be presented in the same order as the chapters in this thesis.

Work presented in Chapter 2 and M. Moor*, M. Horn*, et al. "Topological Autoencoders". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7045–7054: Bastian Rieck initiated the project and introduced Max Horn and Michael Moor to the concepts of persistent homology for the characterization of manifolds. M.M. M.H. and B.R. jointly refined the project direction and thought of ways how to integrate the computations of persistent homology into autoencoder architectures. B.R. and M.M. implemented proofs for the applicability of persistent homology to the level of subsets of the dataset. B.R. performed the empirical convergence rate study for Hausdorff distances. M.M. designed the motivational Spheres dataset. M.H. conceptualized the topological loss term together with B.R., whereas M.M. derived its gradient. M.H. implemented much of the code base, provided the initial project structure and ran most of the initial experiments. M.M. implemented multiple neural network architectures, B.R. and M.M. implemented the measures for non-linear dimensionality reduction. B.R. wrote the code for the computation of persistent homology and the code for several experiments. The experiments for the revision of the paper where run mostly by M.M.. B.R. and M.M. wrote most of the initial manuscript, whereas M.H. helped in its refinement and revision. Karsten Borgwardt and B.R. supervised the project. All authors contributed to the final version of the manuscript.

Work presented in Chapter 3 and C. Weis*, M. Horn*, B. Rieck*, et al. "Topological and kernel-based microbial phenotype prediction from MALDI-TOF mass spectra". *Bioinformatics* 36, 2020, pp. i30–i38: Caroline Weis, Max Horn and Bastian Rieck jointly initiated the project, as part of a greater collaboration and research direction initiated by Adrian Egli and Karsten Borgwardt. C.W. implemented the complete `MALDIquant` preprocessing pipeline and took care of harmonizing the datasets. Further, C.W. provided important insights over the properties of MALDI-TOF spectra and which had an impact on the design of the topological preprocessing and the PIKE kernel. Aline Cuénod and A.E. are clinical collaborators which gathered the data and provided support with regard to clinical relevance. B.R. derived the theoretical foundations for the topological preprocessing and the PIKE kernel. Further, C.W. and B.R. implemented the logistic regression baseline and ran experiments for it. M.H. suggested the utilization of a GP and the focus on rejecting OOD samples. M.H. and B.R. implemented the PIKE kernel. B.R. implemented the topology based preprocessing. C.W., M.H. and B.R. jointly wrote the software package used to run the experiments and process the MALDI-TOF spectra. The initial manuscript was written by C.W. and B.R. M.H. wrote the section on Gaussian Processes and supported writing the discussion of results. The work was supervised by K.B. All authors contributed to the refinement of the manuscript.

Work presented in Section 4.3 and M. Moor*, N. Bennet*, D. Plecko*, M. Horn*, et al. "Predicting sepsis in multi-site, multi-national intensive care cohorts using deep learning". Preprint. 2021. arXiv: `2107.05230 [cs.LG]`: Michael Moor, Nicolas Bennet, Drago Plecko and Karsten Borgwardt conceived the study. Nicolai Meinshausen, Peter Bühlmann and K.B. supervised the study. M.M., N.B., D.P., M.H., B.R., P.B., K.B. designed the experiments. N.B., D.P. performed the cleaning, harmonisation and label annotation. M.M., M.H. implemented the filtering and feature extraction. M.H., M.M implemented the deep learning models. M.M., N.B., D.P. implemented the non-deep ML models. N.B. implemented the clinical baselines. M.M. designed the encounter-focused evaluation. M.M., Bastian Rieck implemented encounter-focused evaluation plots. B.R., M.M. implemented and designed the performance plots. M.H., B.R. implemented the Shapley value calculation. B.R. designed, implemented, and performed the Shapley value analysis. M.M. ran the internal and external validation experiments for all methods. M.M. ran the hyperparameter search of the deep learning models and LightGBM. B.R. ran the hyperparameter search of Logistic regression. N.B. investigated different feature sets. M.M. implemented and ran the max pooling strategy. M.M. designed the pipeline overview figure. D.P., B.R. designed the data harmonisation figure. N.B. designed the risk score illustration and the study flow chart. D.P. devised the dataset table. P.B. and K.B. advised on algorithmic modelling, statistical interpretation and evaluation. All authors contributed to the interpretation of the findings and to the writing of the manuscript.

Work presented in Chapter 5 and M. Horn*, E. D. Brouwer*, et al. "Topological Graph Neural Networks". In: *International Conference on Learning Representations*. 2022: Bastian Rieck conceived the study, B.R. and Karsten Borgwardt supervised the study. Max Horn and Edward De Brower refined the initially conjectured setup for compatibility with general GNN architectures. M.H. implemented the torch C++ kernel used to compute the persistent homology on graphs based on a python implementation of the union find algorithm from E.D. and B.R. M.H. and E.D. implemented the codebase for training models and ran most of the experiments involving training of neural networks in the publication. B.R. ran experiments for the WL and persistence homology based methods. E.D. designed the synthetic datasets used in the motivational example and ran the associated experiments for neural networks. M.H. proposed the design of the static network variant, E.D. and M.H. jointly implemented the required changes and rand associated experiments. B.R. contributed the theory in forms of theorems and proofs, both M.H. and E.D.

contributed to the refinement of the theory. B.R. wrote most of the theoretical sections of the paper, the introduction to persistent homology in the supplements and the description of related work on persistent homology. M.H. and E.D. jointly wrote the experimental section and conclusion. Michael Moor provided support in discussions on the experimental setup and helped in the refinement of the final text. The main text of the publication was revisited multiple times in order to include reviewer feedback, where B.R. and E.D. took care of most rewriting and additional experiments. K.B. and Y.M. assisted in the refinement of the overall story, contributed to the text and provided funding for this endeavour.

Work presented in Chapter 6 and L. O'Bray*, M. Horn*, et al. "Evaluation Metrics for Graph Generative Models: Problems, Pitfalls, and Practical Solutions". In: *International Conference on Learning Representations*. 2022: The idea for the project was initially though of by Max Horn, Bastian Rieck and Karsten Borgwardt. M.H. B.R and Leslie O'Bray wrote the initial draft of the manuscript together. M.H. implemented the training pipeline for graph generative models, the pipeline for the perturbation of graphs and ran the corresponding experiments. B.R. contributed all of the theoretical results and implemented the code for MMD and the Gaussian, Laplacian and Linear kernels. L.O. took over the project fully after initial efforts by M.H. and ran the MMD evaluations on perturbed and generated graphs together with M.H. Further, L.O. implemented an accelerated MMD variant, the total variation kernel and ran all analysis experiments and created all plots of the paper. B.R. and L.O. additionally explored the creation of topology based scoring functions yet decided to move their efforts to a separate publication. M.H. gave advice on the refinement of the manuscript, wrote some minor sections of the paper and helped revise the final manuscript. B.R. and K.B. jointly supervised this project, all authors contributed to the manuscript.

Part I

Dimensionality Reduction

This part of the thesis concentrates on developing approaches for dimensionality reduction using aspects from topoogy.

2 Preserving Underlying Structure using Multi-Scale Topology

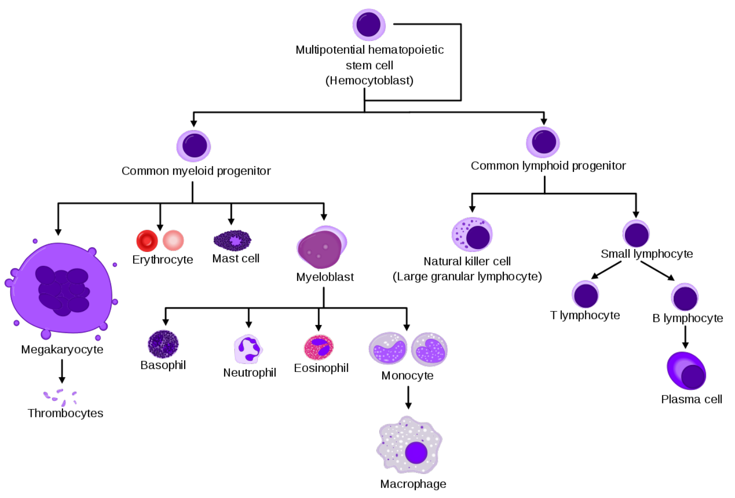

Multipotential hematopoietic stem cell (Hemocytoblast)
Common myeloid progenitor
Common lymphoid progenitor
Megakaryocyte
Thrombocytes
Erythrocyte
Mast cell
Myeloblast
Basophil
Neutrophil
Eosinophil
Monocyte
Macrophage
Natural killer cell (Large granular lymphocyte)
Small lymphocyte
T lymphocyte
B lymphocyte
Plasma cell

Figure 2.1: Visualization of cell differentiation during hematopoiesis. Image by A. Rad and M. Häggström, distributed under a CC-BY-SA 3.0 license[1].

Often we don't know the underlying structure of data we would like to explore and understand. Should we assume instances in the data are clustered? Are there hierarchical or nested structures present? One core issue here is that these questions often require deciding on a scale at which they should be answered. For example, how many points must be grouped to assume a cluster is present? How close do these have to be together? Many approaches in dimensionality reduction require the selection of hyperparameters which implicitly decide over the answers to the above questions [44]. This is especially problematic in biological research where modern methods such as single-cell sequencing allow the derivation of large amounts of high-dimensional data. What if there is no appropriate scale that we should consider as there are hierarchies of similarities between subpopulations? These situations arise naturally in biology as cells are not created completely distinct from each other, but undergo many steps of differentiation before they take on their final

[1]https://commons.wikimedia.org/wiki/File:Hematopoiesis_simple.svg

form. Thus, each dataset would not only contain clusters of different cells but also cells that are on a trajectory of transitioning from one less specialized cell type to a more specialized one.

A particularly striking example is the formation of our blood cells, hematopoiesis, which is visualized in Figure 2.1. Here we see the clear presence of hierarchical and multi-scale structures that can arise in biological systems. In this particular example, we would expect cells derived from the same precursor to show similar gene expression patterns, compared to cells that have branched off much earlier in the hierarchy. A further example from biology is the hierarchical structure of cells forming tissues, tissues forming organs, and organs forming the overall organism. Here we would also expect that cells of the same tissue are more similar to cells of different tissues or organs. For example, it seems reasonable, that cells present in the brain show similar gene and protein expression as they have to deal with the same environment compared to cells in the liver or kidney where completely different environments are present [236].

This chapter introduces the problem of dimensionality reduction and shows how to use techniques from persistent homology to derive lower-dimensional representations of data, which preserve multi-scale relationships. It is largely based on the following published work:
M. Moor*, M. Horn*, et al. “Topological Autoencoders”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7045–7054

2.1 Manifolds

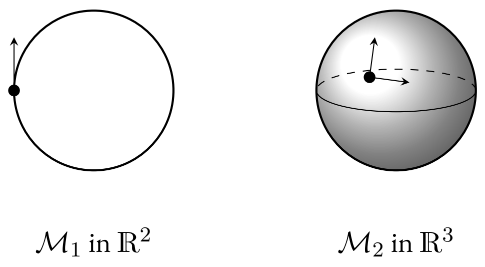

$\mathcal{M}_1$ in $\mathbb{R}^2$
$\mathcal{M}_2$ in $\mathbb{R}^3$

Figure 2.2: Examples of manifolds embedded into higher dimensional spaces. Arrows indicate the tangents to the manifold, that serve as coordinates in the approximately Euclidean neighborhood the indicated point.

To talk about dimensionality reduction in a principled and formal manner it makes sense to first understand the concept of a manifold. Given our knowledge of topological spaces as introduced in Section 1.2, a manifold requires additional criteria on top of the definition of a topological space. In particular, a d-dimensional manifold is a structure that *locally* looks like d-dimensional Euclidean space. The most obvious simplified example one could give for a real-world manifold that we all live on is earth (for simplicity of this example, we assume that we live merely in 2D space instead of 3D space). While most people are reasonably convinced that our planet is a spherical structure, for most purposes this is completely irrelevant and the earth could simply be considered locally 2-dimensional as there are 2 axes on which we can move — the north-south and the east-west axis. The earth thus appears to us *locally* as a 2-dimensional plane. Solely when we start moving very long distances on this plane or try to fly into space does the actual 3D structure of the sphere become relevant.

A lower-dimensional example of the above can be built by assuming a creature that lives on a line forming a circle. For this creature, there exist only two directions forward and backward, thus the space it lives on can be assumed to be the 1-dimensional line. Yet, while space seems locally 1D, if the creature walks long enough in one direction, it will reach its previous position at some point in time, uncovering the macroscopic perspective that the space it lives on is actually "bent". This notion of space being bent is closely related to what students of manifolds refer to as *curvature*.

Let us take a step back and formularise our insights.

Definition 2.1 (Manifold)**.** A d-manifold $\mathcal{M}_d$ is a topological space, where each point has a neighborhood that is homeomorphic to an open subset of d-dimensional Euclidean space $\mathbb{R}^d$.

Thus for each point we require the existence of a homeomorphism, i.e. a transformation which "does not tear or glue space", from the neighborhood of each point to the d-dimensional Euclidean space. For the sake of clarity, we leave out some of the more formal requirements on topological manifolds and their classification [2].

2.1.1 The Manifold Hypothesis

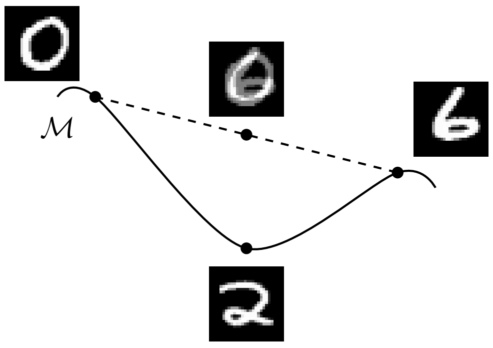


Figure 2.3: Visualization of interpolations on a manifold using MNIST [130] digits. The manifold $\mathcal{M}$ is denoted with a continuous line, whereas linear interpolation between instances of the manifold is denoted with a dashed line.

[2]For example, we strictly require the topological space to be a Hausdorff space which would imply that for any two points there exist neighborhoods that are disjoint. All requirements are satisfied in $\mathbb{R}^d$ though.

The manifold hypothesis is one of the most widespread assumptions present in machine learning. It assumes that the data we work with typically live on a lower dimensional manifold $\mathcal{M}_{d_1}$ embedded into the higher dimensional input space on which we operate, $\mathbb{R}^{d_2}$. Thus the intrinsic dimensionality of a problem at hand is considered much lower than the dimensionality of the input domain, i.e. $d_1 \ll d_2$.

Let us take an example from the computer vision domain. If our goal is to classify natural images in $\mathbb{R}^{w \times h}$ based on their contents, then it is clear that not any random input would be considered a natural image. Thus our data, for example, a collection of cat pictures, is derived from a subspace $\mathcal{D} \subset \mathbb{R}^{w \times h}$ of all possible inputs, and thus its intrinsic dimensionality and complexity might be completely different from the input dimensions $w \times h$. Moreover, if we simply upsample w to $2 \cdot w$, this increases the dimensionality of the input space, but does not necessarily change the complexity or intrinsic dimensionality of the problem.

A further perspective arises by considering interpolations of images. While linear interpolation creates a mixture of two images, these images can often not be considered natural images. Ideally, we would want that interpolations between two images would still be considered natural images. This could be done by smoothly removing and adding features between the interpolated images. If we would consider two pictures of a meadow, one with a tree on it and one with no trees, then we would not want the tree to become more and more transparent when interpolating as transparent trees are not natural. Instead, we would want to take a path of increasingly smaller trees (or better *younger* trees) until there is no tree to be seen anymore. An example on the MNIST [130] digit manifold is shown in Figure 2.3. As in the tree example, the linear interpolation between two MNIST digits would also not be considered a real digit. In contrast, interpolation along the manifold of MNIST digits, could lead to morphing our starting point of the digit 0 first into a 2 (and thus gradually removing one side of the digit) and then into a 6 (by moving the line to the other side of the image and increasing the loop in the bottom).

2.1.2 Manifold Learning vs. Dimensionality Reduction

While one could see manifold learning as the idealized approach for deriving underlying structures from unstructured high-dimensional data, it is an inherently difficult task. Instead, most research to date focuses on the topic of dimensionality reduction. Dimensionality reduction is less constrained than manifold learning as it does not actually try to recover the underlying manifold of the data, and thus does not have the additional smoothness and connectedness constraints which arise due to the required local Euclidean structure. Instead, dimensionality reduction allows neighborhoods of points to follow different geometries than those defined for a manifold. It is thus a more flexible approach concerning the mapping between the high-dimensional and low-dimensional space, yet removes some of our ability to interpret the lower dimensional representation.

2.2 Dimensionality Reduction

In general, we can phrase dimensionality reduction as an unsupervised regression task. The goal is to find a mapping $f : \mathbb{R}^d \to \mathbb{R}^k$ from a d-dimensional input space $x_i \in \mathbb{R}^d$ to a lower-dimensional ($k < d$) representation $z_i \in \mathbb{R}^k$ based on a provided dataset $\mathcal{D} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$. This low-dimensional representation is often referred to as *embedding*. Dimensionality reduction can be used to tackle many different tasks. As illustrated in the introduction of this section, it might be used as a means to *visualize* the data in a space where we can have a better (intuitive) understanding of it (i.e. by reducing the dimensionality to 1, 2, or 3 dimensions). Further, it can be utilized to regularize machine learning problems by reducing the number of input dimensions provided to a machine learning model. Finally, dimensionality reduction can be used for feature selection / fusion to derive lower dimensional representations of the data where features are “more meaningful”.

As the problem of dimensionality reduction without further constraints is inherently ill-defined, the key question in dimensionality reduction arises naturally: Which mapping f of the potential set of mappings $f \in \mathcal{F}$ should be preferred over others and how is this class of potential mapping defined? This question represents the dimensionality reduction version of a similar dilemma of machine learning, the bias-variance tradeoff[3].

2.2.1 Linear Dimensionality Reduction

In the case of linear dimensionality reduction, we constrain f to be of the form $f(x) = W^\top x$, with $W \in \mathbb{R}^{d \times k}$. The most common approach to linear dimensionality reduction is Principal Component Analysis (PCA), where besides the linear constraint, we additionally try to minimize the mean squared distance between the inverse mapping applied to the lower dimensional representation $f^{-1}(z)$ and the input data x. To ensure the uniqueness of the result, an additional orthogonality constraint is applied to W, as we could otherwise simply rescale both the lower dimensional representation as well as W itself.

[3]Here the question is how to balance the tradeoff between simplifying assumptions in the machine learning model, i.e. restricting the class of potential mappings $\mathcal{F}$, with its potential to overfit, i.e. how high is the variance of the model’s predictions when trained on independent samples from the same distribution.

2.2.2 Non-Linear Dimensionality Reduction

The class of non-linear dimensionality reduction methods allows much higher flexibility for the function f, such that the function class is much larger. Thus the number of potential methods is also significantly higher. I provide a short overview of prominent approaches below.

Multidimensional Scaling (MDS): Optimizes the positions of instances in the lower dimensional space such that their distances approximate those of the input data. The mapping f is thus non-parametric and only defined for points that were provided at the start of the algorithm.

Isomap [228]: Application of MDS to geodesic distances which are approximated using only distances of nearest neighbors. The whole geodesic distance matrix is computed efficiently by exploiting the Floyd-Warshall shortest path algorithm.

t-distributed stochastic neighborhood embedding (t-SNE) [148]: Optimizes the points in the lower dimensional space such that a constructed probability distribution of the original data and the reduced dimensional data are close in terms of the KL-divergence. Here, the probabilities in the data space and the reduced dimensional space are computed differently, in the former case using a Gaussian distribution and in the latter case using a Cauchy distribution. t-SNE also does not provide a parametric map between data and reduced dimensional space.

Uniform Manifold Approximation and Projection (UMAP) [153]: A method based on theory from Riemannian geometry and algebraic topology. Based on a set of assumptions (uniform distribution on a manifold, locally constant Riemannian metric and local connectedness of the manifold) the method models the data using a fuzzy topological structure and searches for a low-dimensional projection which is most aligned with the fuzzy topology of the data.

Autoencoders [87]: Autoencoders learn a parametric mapping f between data and reduced dimensional space which is trained together with an inverse mapping g that maps back to the data space. The model is then trained using a reconstruction loss between the input x and $g(f(x))$. Both f and g are often deep neural networks, which dependent on the type of data being processed can either be conventional MLPs, CNNs or even RNNs.

2.3 Persistent Homology in $\mathbb{R}^d$

Despite having previously discussed notions of manifolds and their relevance to the problem of dimensionality reduction, it is still not clear how a manifold can be appropriately characterized. It

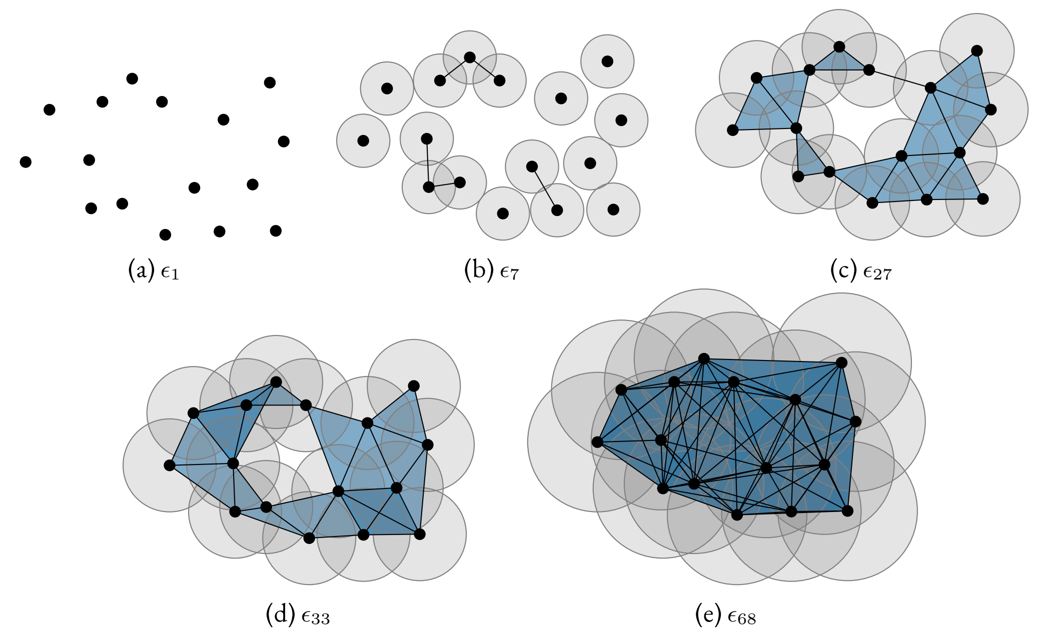


Figure 2.4: The Vietoris–Rips complex of a point cloud at different scales ϵ_1, ϵ_7, ϵ_{27}, ϵ_{33} and ϵ_{68}. The simplicial complex changes as a function of the distance threshold ϵ leading to more and more nodes to be considered connected.

turns out, that here topology and our previously introduced notions of persistent homology are of high utility. In general, the requirement of local connectedness of manifolds ties back to the core perspective that topology uses to examine topological spaces via homology — the presence and character of holes. Two-dimensional manifolds such as spheres and tori (i.e. doughnut shapes) behave very differently and cannot be smoothly formed into each other. While the sphere is simply connected, i.e. any loop on the sphere can be smoothly transformed into a point, this is not the case for the torus as some loops cannot be reduced to a point without the necessity of gluing or cutting. The insight that homology represents a good characteristic for a manifold dates back to Henri Poincaré who showed that every simply connected and closed 2-manifold is equivalent to the 2-sphere and famously conjectured that the same is true for simply connected 3-manifolds to 3-spheres. Further, research in persistent homology has shown that the representations derived from persistent homology are good characterizations of the manifold structure [69]. In the following, we thus explain how the notions from persistent homology can be applied to spaces where the connectivity of points is not predefined, such as point clouds.

The Vietoris–Rips Complex When working with data in a high-dimensional space $\mathbb{R}^d$, we don't have any notion of connectivity, making notions of topology relatively ill-defined as in this case the neighborhood of a point is not known. Generally, it is not clear at which distance or scale two points should be considered connected. In the framework of persistent homology, we avoid this problem to some degree by constructing a filtration based on some properties of the input space, which defines connectedness of neighborhoods in a nested manner. We would thus start with no data points being connected, and end with all of them being connected. While there are multiple ways how such a filtration could be defined to give rise to a simplicial complex, the most widespread is the utilization of the Vietoris–Rips complex [243]. The Vietoris–Rips complex can be constructed on any metric space and thus also on our high-dimensional space $\mathbb{R}^d$ if we assume a metric such as the Euclidean metric.

The Vietoris–Rips complex VR is constructed by forming a simplex for any finite set of points with a maximum diameter of ϵ. In practice this means we consider two points connected if they are maximally within ϵ-distance of each other. The Rips–Graph is a neighborhood graph used to compute the Vietoris–Rips complex and can be defined as $\mathcal{R}_\epsilon = (V, E)$, with

$$\begin{aligned} V &:= \{1, 2, 3, \dots\} \\ E &:= \{(u, v) \mid \text{dist}(x_u, x_v) \leq \epsilon\} \end{aligned}$$

where $x_u, x_v \in \mathbb{R}^d$.

Accordingly, we can define a sublevel set filtration function as

$$f(x_i, x_j) = \text{dist}(x_i, x_j) \tag{2.1}$$

which would lead to a sequence of simplicial complexes for which $\emptyset = \mathrm{K}^{(0)} \subseteq \cdots \subseteq \mathrm{K}^{(n)}$ for $\epsilon_0 \leq \cdots \leq \epsilon_n$. Typically, it is assumed that all vertices (in our case data points in the high dimensional space) are created at the beginning of the filtration, which can for example be implemented by assigning them the filtration value of $\epsilon_0 < f(x) < \epsilon_1$. The overall process of constructing a filtration via the Vietoris–Rips complex can be understood as increasing the neighborhood in which points would be considered connected until we in the end have a fully-connected / complete graph. A visualization hereof is depicted in Figure 2.4. The computation of filtrations using the Vietoris–Rips complex can also be extended to higher dimensional structures such as triangles, where we for example would define the filtration value using the maximum of the filtration values of all involved edges to preserve the properties of filtrations:

$$f(x_i, x_j, x_k) = \max\{\text{dist}(x_i, x_j), \text{dist}(x_j, x_k), \text{dist}(x_k, x_i)\} \tag{2.2}$$

2.4 Topological Autoencoders

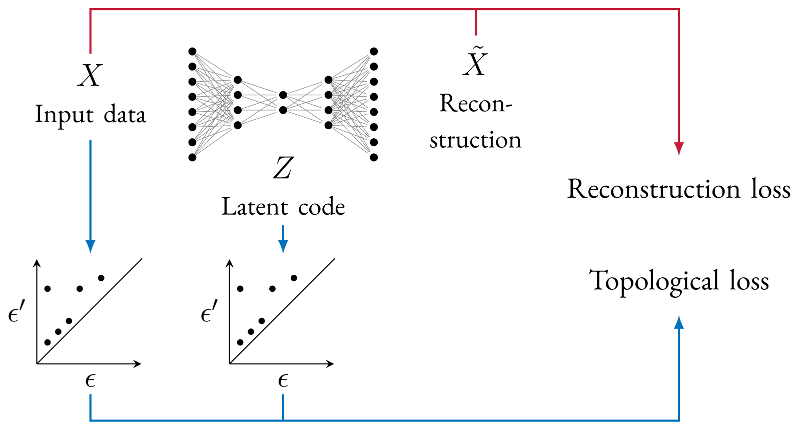


Figure 2.5: Overview of the Topological Autoencoder method. The topological autoencoder consists of two components 1. a regular autoencoder, which is trained to reconstruct the input based on a lower dimensional (represented in red) and 2. a topological loss term that incentivises the neural network to *preserve* the topological structure of the input data (represented in blue) . The topological loss is constructed using persistence diagrams derived from the Vietoris–Rips complex calculated from the input and latent data points.

In the following section, I will present and discuss Topological AutoEncoders (TopoAE), a non-linear dimensionality reduction approach which strives to preserve the multi-scale topological structure of input data in a latent space of reduced dimensionality. The core idea of the approach is to harness the power of autoencoders to construct highly non-linear representations of reduced dimensionality and combine them with the manifold characterizing properties of persistent homology [68] to ensure that the low-dimensional representations preserve topological structure.

The model computes a low dimensional representation of an input batch similar to a standard autoencoder, yet additionally computes topological signatures from the batch of input elements and the batch of their latent embeddings. These signatures are used to construct an additional loss term that aims to match the topological signatures of input and latent space. The topological loss and the reconstruction loss of the autoencoder are combined to form the final training loss. An overview of the approach is shown in Figure 2.5. TopoAE relies on recent developments for propagating gradients through the computation of persistent homology [98] and presents a general way how to constrain autoencoders to preserve topological structure in the input space, when computing lower dimensional latent representations.

Notation Given a collection of data X with $|X| = m$ we denote the matrix of all pairwise distances between elements of X as $\mathbf{A}^X$, where $\mathbf{A}^X_{i,j} = \mathrm{dist}(x_i, x_j)$. Further, we use the notation $\mathbf{A}[((i,j),(j,k))] := (\mathbf{A}_{i,j}, \mathbf{A}_{j,k})$ to indicate subsetting of the distance matrix. Additionally, we denote the Vietoris–Rips complex (see Section 2.3 for an introduction) constructed using a threshold ϵ as $\mathrm{VR}_\epsilon(X)$, and represent the complete filtration from the smallest distance in the dataset $\epsilon_0 = \min_{\{x_i,x_j\}\in X} \mathrm{dist}(x_i,x_j)$ to the largest distance present $\epsilon_n = \max_{\{x_i,x_j\}\in X} \mathrm{dist}(x_i,x_j)$ as $\mathrm{VR}(X) = (\mathrm{VR}(X)_{\epsilon_0}, \dots, \mathrm{VR}(X)_{\epsilon_n})$, where $\mathrm{VR}(X)_{\epsilon_0} \subseteq \cdots \subseteq \mathrm{VR}(X)_{\epsilon_n}$ and $\epsilon_0 \leq \cdots \leq \epsilon_n$. Further, we denote the computation of persistent homology on such a complex as $\mathbf{ph}(\mathrm{VR}(X))$. The computation returns tuples of *persistence diagrams* and *persistent pairings* for all considered dimensions of topological features $((\mathcal{D}_0, \mathcal{D}_1, \dots), (\pi_0, \pi_1, \dots))$.

Each d-dimensional persistence diagram contains tuples of the form (ϵ, ϵ'), where ϵ denotes the distance at which a topological feature is created and ϵ' denotes the distance at which a topological feature is destroyed. For a 0-dimensional persistence diagram, for example ϵ' corresponds to the distance at which two connected components merge into a single one. In the simplest case, this could represent two vertices being connected via an edge. Then the persistence diagram would contain a tuple with the value at which the *older* vertex is created and the value at which the edge is introduced as its destruction point. The persistence pairings on the other hand contain the indices (i,j) which correspond to the simplices $s_i, s_j \in \mathrm{VR}_\epsilon(X)$ that created and destroyed the topological feature. In our simple example it would contain the index of one of the vertices and the index of the edge which destroyed it.

Selection of indices from persistence pairings Our methods rely on selecting indices from the persistence pairing and mapping them back to the distance between two vertices. For 0-dimensional features, it is in practice sufficient to consider the indices of edges that always correspond to the destroyer simplex of a persistent pairing. This makes sense, as we generally assume all vertices to be created at the beginning of the filtration and thus their creation values would be considered the same. For the 0-dimensional persistence diagram these edges correspond to edges in the minimum spanning tree of the dataset. Based on preliminary experiments, this work limits itself to the utilization of 0-dimensional topological features. The applicability of the method to higher dimensional topological features is treated in further detail in the discussion.

Method description We consider a mini-batch of data X sampled from a data space $\mathcal{X}$ with $|X| = m$, as a point cloud. Further, we define an autoencoder to be the composition of two mappings $h \circ g$ where $g : \mathcal{X} \to \mathcal{Z}$ represents the *encoder* and $h : \mathcal{Z} \to \mathcal{X}$ the *decoder* and $\mathcal{Z}$ represents the latent space of reduced dimensionality. Further, we denote the latent embeddings of our input as $Z := g(X)$. When computing the forward pass of the autoencoder, we compute persistent homology both for the original mini-batch input $(\mathcal{D}^X, \pi^X) := \mathbf{ph}(\mathrm{VR}(X))$ and the latent representation of reduced dimensionality $(\mathcal{D}^Z, \pi^Z) := \mathbf{ph}(\mathrm{VR}(Z))$.

The 0-dimensional persistence diagram essentially represents a subset of the distance matrix such that we write $\mathcal{D} \simeq \mathbf{A}[\pi]$. Thus informally, the persistent homology computation can be seen as the selection of topologically relevant edges of the Vietoris–Rips complex. We denote the topological regularisation term as $\mathcal{L}_t(\mathbf{A}^X, \mathbf{A}^Z, \pi^X, \pi^Z)$. It accounts for the degree of mismatch between the two persistence diagrams that is introduced by the non-linear dimensionality reduction step of the autoencoder. We include this term in the overall loss term of the network used during training

$$\mathcal{L} := \mathcal{L}_r + \lambda \mathcal{L}_t$$

where $\lambda \in \mathbb{R}$ controls the strength of the regularisation and $\mathcal{L}_r$ corresponds to the reconstruction loss of the autoencoder.

Topological loss term A possibility to ensure the latent space mimics the topology of the input space, would be to directly calculate a loss based on the selected distances in both spaces. Yet this would not lead to informative gradients for the autoencoder, as it only compares the topological features between both spaces, which could lead to the differences between completely unrelated edges being minimized as the identity (which node is actually responsible for creating or destroying the topological feature) gets lost. While this makes sense in the comparison of completely unrelated spaces, we are in a scenario where more information is available. In particular, we know exactly which element in the input space, gets transformed into which latent representation.

One could thus use the intersection of edges selected in both persistence pairings, yet this could lead to problems at the beginning of the optimization procedure where only very few edges would actually be selected in both input and latent space. In particular, if we assume that distances between points are distorted to become completely random via the mapping g, the expected value of matched edges is only 1 independent of the batch size. This would lead to extremely high variance in the topological regularization term as only a single edge might be used to characterize the topological alignment of the two spaces.

Instead, we compute the topological loss term based on the union of both selected edges. In particular,

$$\mathcal{L}_t := \frac{1}{2m} \left\| \mathbf{A}^X[\pi^X \cup \pi^Z] - \mathbf{A}^Z[\pi^X \cup \pi^Z] \right\|^2 \tag{2.3}$$

where we define the union such that elements present in both persistence pairings will be present twice in the union. The factor $2m$ is added to ensure the loss is on comparable scales for different selections of the batch size m. This formulation takes at least $|X| = m$ distances into account (when the persistence pairings are perfectly matched). The loss can be viewed as a generalization of the loss introduced in [98], yet constructed with a different goal in mind. While our loss is zero when the "topologically relevant" distances from X and Z align perfectly, it is *not* uniquely minimized in this case, i.e. there exist scenarios where $\mathcal{L}_t = 0$ yet, $\pi^X \neq \pi^Z$.

2.5 Theoretical Considerations

This section inspects the theoretical aspects of TopoAE. First, we verify that it is possible to compute a gradient through the topological loss term, afterwards, we check if computing persistent homology on the level of a mini-batch has similar theoretical guarantees as performing the computation on the whole dataset.

2.5.1 Differentiability

The differentiability of persistent homology is based on the assumption that all pairwise distances in the data X are unique. This is a reasonable assumption to make if the data live on the domain of real numbers $\mathbb{R}^d$ as then the probability of getting two times the same distance would have Lebesgue measure 0. We will now show we can compute a gradient through the computation of persistent homology given the above assumption.

Theorem 2.1 (Differentiability of Topological Loss). Let X be a data sample, $f_\theta : \mathcal{X} \to \mathcal{Z}$ a differentiable encoding function with continuous parameters θ and dist a differentiable distance metric. If all pairwise distances of X are unique, i.e. there do not exist two distinct pairs of data points x_i, x_j and x_k, x_l such that $\text{dist}(x_i, x_j) = \text{dist}(x_k, x_l)$, then the topological loss term $\mathcal{L}_t$ is differentiable with respect to θ.

Proof. Let $\epsilon_{\rho(1)}(\theta)$ denote a distance in the pairwise distance matrix $\mathbf{A}_\theta^Z$ of $Z := f_\theta(X)$ with regard to the sorting permutation ρ induced by **ph**, i.e. $\epsilon_{\rho(1)}(\theta) < \epsilon_{\rho(2)}(\theta) < \cdots < \epsilon_{\rho(m)}(\theta)$, where $m = |X|(|X| - 1)/2$. By assumption, all ϵ values are unique, thus there exists a neighborhood h around θ such that the ordering does not change given changes of θ within this neighborhood. Thus,

$$\exists \epsilon > 0 \forall h : |h| < \epsilon \implies \epsilon_i(\theta + h) \neq \epsilon_j(\theta + h)$$

and

$$\epsilon_{\rho(1)}(\theta + h) < \epsilon_{\rho(2)}(\theta + h) < \cdots < \epsilon_{\rho(m)}(\theta + h).$$

This further, implies that for this sufficiently small change of the parameters, the induced filtration also does not change, such that the persistence pairings π remain constant, i.e.

$$\pi^Z(\theta) = \pi^Z(\theta + h)$$

That allows us to threat the subsetting operation as constant when computing the derivative. Thus also the union $\pi = \pi^X \cup \pi^Z$ of both persistence pairing can be treated as constant, as π^X does not change dependent on θ by definition.

$$\begin{aligned}\frac{d}{d\theta} &= \frac{d}{d\theta}\left(\frac{1}{2}\left\|\mathbf{A}^X[\pi] - \mathbf{A}^Z[\pi]\right\|^2\right)\\ &= -\boldsymbol{\rho}^T\left(\frac{d\mathbf{A}^Z[\pi]}{d\theta}\right)\end{aligned}$$

where $\boldsymbol{\rho} := \left(\mathbf{A}^X\left[\pi^X\right] - \mathbf{A}^Z\left[\pi^X\right]\right)$. $\square$

2.5.2 Stability

While it has been shown that the computations of persistent homology are stable with respect to small perturbations [54], it is not yet clear if this also applies to individual subsets of the input data. This is of particular interest as current neural network architectures are typically trained using Stochastic Gradient Decent (SGD) which relies on stochastic updates to the parameters which were computed on subsamples of the data. In the following we will formularise the problem in terms of distances to the persistence diagram of the overall dataset and show that persistent homology computations on a subset of the data is close to the diagram of the overall dataset in terms of the bottleneck distance. We start with some definitions.

Definition 2.2 (Hausdorff Distance)**.** The Hausdorff distance $\mathrm{d_H}$ between two persistence diagrams $\mathcal{D}$ and $\mathcal{D}'$ is defined as

$$\mathrm{d_H}(\mathcal{D}, \mathcal{D}') := \max\left\{\sup_{x\in\mathcal{D}}\inf_{y\in\mathcal{D}'}\operatorname{dist}(x,y),\ \sup_{y\in\mathcal{D}'}\inf_{x\in\mathcal{D}}\operatorname{dist}(x,y)\right\}, \tag{2.4}$$

where dist refers to a base distance such as the Euclidean distance.

The Hausdorff Distance thus represents the maximal distance that any element in $\mathcal{D}$ would need to travel in order to reach an element of $\mathcal{D}'$. This allows also non-unique matchings, i.e. that multiple elements in $\mathcal{D}$ would consider the same element in $\mathcal{D}'$ as their closest point. In contrast, the Bottleneck Distance requires a bijection between points of the diagrams and thus represents an upper bound to the Hausdorff distance.

Definition 2.3 (Bootleneck Distance)**.** The bottleneck distance between $\mathrm{d_b}$ two persistence diagrams $\mathcal{D}$ and $\mathcal{D}'$ is defined as

$$\mathrm{d_b}(\mathcal{D}, \mathcal{D}') := \inf_{\eta:\mathcal{D}\to\mathcal{D}'}\sup_{x\in\mathcal{D}}\|x - \eta(x)\|_\infty, \tag{2.5}$$

where $\eta : \mathcal{D} \to \mathcal{D}'$ denotes a bijection between the points of the two diagrams and $\|\cdot\|_\infty$ refers to the L_∞ norm.

It refers to the maximal distance between two matched points (i.e. points that are mapped to each other via the bijection ν), for the best possible matching between the two diagrams (the matching which achieves the lowest distance). First, we show that we can upper bound the probability that the bottleneck distance between the persistence diagram of the whole dataset $\mathcal{D}$ and a subsample thereof $\mathcal{D}^{(m)}$ exceeds a value of ϵ by the probability of the Hausdorff distance exceeding 2ϵ on the original input data.

Theorem 2.2 (Probabilistic Bound Bottleneck Distance)**.** Let X be a point cloud of cardinality n and $X^{(m)}$ be one subsample of X of cardinality m, i.e. $X^{(m)} \subseteq X$, sampled without replacement. We can bound the probability that two resulting persistence diagrams $\mathcal{D}^{(m)}$ and $\mathcal{D}$, exceed a threshold in terms of the bottleneck distance as

$$\mathbb{P}\left(\mathrm{d_b}\left(\mathcal{D}^X, \mathcal{D}^{X^{(m)}}\right) > \epsilon\right) \leq \mathbb{P}\left(\mathrm{d_H}\left(X, X^{(m)}\right) > 2\epsilon\right), \tag{2.6}$$

where $\mathrm{d_H}$ refers to the Hausdorff distance between the point cloud and its subsample.

Proof. Chazal, Silva, and Oudot [46] characterized the stability of persistent homology calculations for finite metric spaces. Applying their theorem to our context[4], we can show that

$$\mathrm{d_b}\left(\mathcal{D}^X, \mathcal{D}^{(m)}\right) \leq 2\,\mathrm{d_{GH}}\left(X, X^{(m)}\right), \tag{2.7}$$

where $\mathrm{d_{GH}}(\cdot,\cdot)$ refers to the Gromov–Hausdorff distance [35, p. 254] of the two spaces and is defined as the infimum Hausdorff distance over all isometric embeddings of the two spaces. In our case we can assume both spaces to have the same metric as one is a subset of the other. By definition of the Gromov–Hausdorff distance we can further upper bound this statement with the Hausdorff distance of a specific isometric embedding $\mathrm{d_{GH}}(X, Y) \leq \mathrm{d_H}(X, Y)$ such that we thus have

$$\mathrm{d_b}\left(\mathcal{D}^X, \mathcal{D}^Y\right) \leq 2\,\mathrm{d_H}(X, Y). \tag{2.8}$$

The final statement arises by rewriting the above inequality in terms of probabilities. □

As m converges to the size of the complete dataset n, we have $\lim_{m\to n} \mathrm{d_H}\left(X, X^{(m)}\right) = 0$.

[4]it is actually valid in a much more generic context then our case of $X^{(m)} \subset X$

2.6 Experimental Evaluation

In the following, we compare the capability of TopoAE to derive lower dimensional representations via non-linear dimensionality reduction while preserving multi-scale topological structure (as quantified by persistent homology computations). For this, we first qualitatively compare TopoAE to other linear and non-linear dimensionality reduction approaches on a synthetic dataset where the multi-scale structure is present by design. We then qualitatively compare the lower-dimensional representations of the approaches on common datasets. Finally, we evaluate all approaches quantitatively using metrics used to quantify the performance of dimensionality reduction approaches. We start by describing the experimental setup.

2.6.1 Experimental Setup

Datasets To explicitly visualize the topology-preserving structure of TopoAE, we created a simulated dataset Spheres where the multi-scale topological structure is known. It contains 10 100-dimensional spheres living in a 101-dimensional space, that are enclosed by one larger sphere, which contains the same number of points as the total of all nested inner spheres. Further, we consider three image datasets MNIST [130], Fashion-MNIST [260] and CIFAR-10, which were selected as real-world images are known to lie on low-dimensional manifolds [131, 184].

Evaluation Metrics To quantitatively assess the performance of all considered models, we utilize a series of quality measures commonly used in non-linear dimensionality reduction literature and additionally include a metric we developed ourselves which can quantify the distributional alignment dependent on a selected scale. We construct this metric as the Kullback–Leibner divergence between density estimates of the input and latent space, based on previous work by Chazal, Cohen-Steiner, and Mérigot [45]. We derive the density estimates at different scales using a Gaussian kernel where we vary the lengthscale parameter $\sigma \in \mathbb{R}_{>0}$ and denote the density estimate for X using the lengthscale σ as $\mathrm{f}_\sigma{}^X(x) := \sum_{y \in X} \exp\left(-\sigma^{-1} \operatorname{dist}(x,y)^2\right)$. We then denote the evaluation metric based on the density estimates of input space X and latent space Z as $\mathrm{KL}_\sigma = \mathrm{KL}\left(\mathrm{f}_\sigma{}^X \parallel \mathrm{f}_\sigma{}^Z\right)$. To avoid confusion, we denote other metrics which quantify the alignment of distances in latent and data space with ℓ. We consider the ℓ-RMSE which quantifies the *root mean square error* between the distances of input and latent representation, the *mean relative rank error* (ℓ-MRRE), *continuity* (ℓ-continuity), and *trustworthiness* (ℓ-trust).

Baselines and Training Procedure We compare TopoAE to several dimensionality reduction techniques PCA, Isomap [228], t-SNE [148], UMAP [153] and a conventional Autoencoder [87]. When running experiments we keep the architecture of the Autoencoder and TopoAE the same to solely quantify the contribution of the topological loss term.

To allow maximum comparability from a qualitative evaluation perspective, we enforce the lower dimensional space to be two-dimensional. This allows us to easily visualize and compare the lower dimensional representations. Evaluation and visualization of the approaches are performed on a held out test split, where we rely on the predefined split for MNIST, Fashion-MNIST and CIFAR-10 and create a random split of 10% of the data for the Spheres dataset. The hyper-parameters of all approaches are optimized with respect to the $\mathrm{KL}_{0.1}$ dimensionality reduction quality metrics on a 15% validation split of the training data. Both autoencoder approaches employ batch normalization [109] and were optimized using Adam [121]. In comparison to all other methods, t-SNE cannot be evaluated on previously unseen test samples as it does not give rise to a parametric embedding function. We thus evaluate t-SNE only on the train split, giving it a slight advantage compared to the other approaches.

2.6.2 Qualitative Evaluation on Synthetic Data

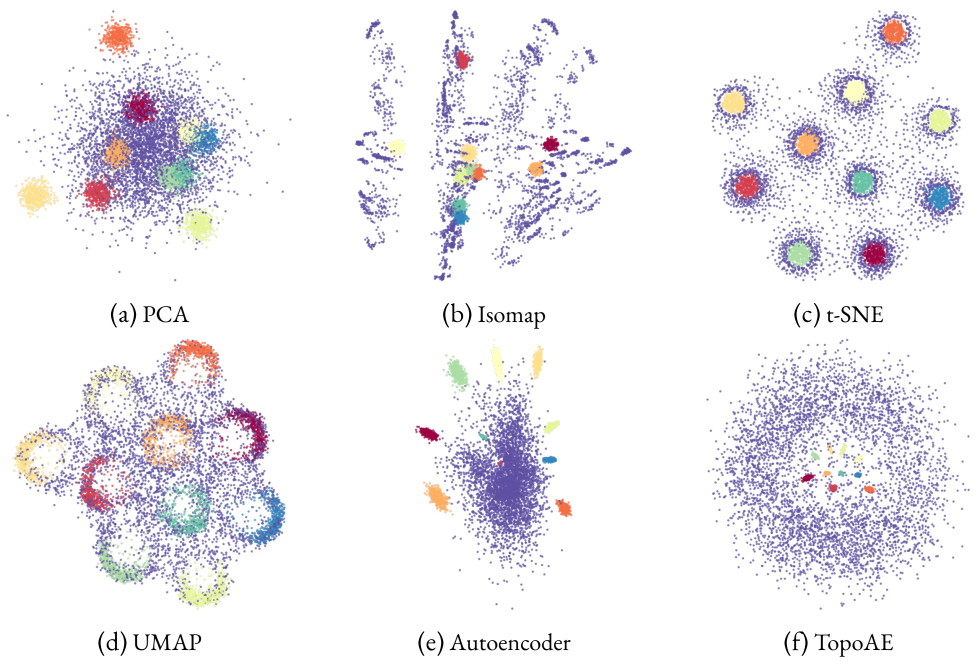


Figure 2.6: Visualization of latent representations derived by different methods of the Spheres dataset. The identity of the individual spheres is color coded. TopoAE is able to represent the nesting relationship present in the data, whereas all other approaches fail to capture the structure.

As shown in Figure 2.6, the preservation of the nested relationships between the inner spheres and the encompassing sphere is not guaranteed for most dimensionality reduction methods, which indicates that this task is particularly difficult if the topological structure is not accounted for. In contrast, TopoAE is able to capture this nesting relationship reliably. We can identify mainly two failure modes in the other approaches. While PCA, Isomap, and autoencoder all treat the surrounding sphere as a kind of noise in between the other spheres or as a cluster itself, both t-SNE and UMAP associate the instances of the outer sphere with individual inner spheres. Interestingly, UMAP – the only other topologically motivated method – tries to capture the structure of the nested spheres by representing them as (half) circles, yet does not capture the relation of these spheres with the surrounding sphere. This highlights how it can be important to capture relationships *across multiple scales* as we do in TopoAE.

It is important to note, that the task is by construction very difficult for many of the comparison methods. This is partially due to them not trying to preserve global structure but instead focusing on the local structure or because their design assumptions are not fulfilled. For example, UMAP assumes the underlying data to be uniformly distributed on a locally connected Riemannian manifold [153], yet in our case, we generate data from multiple disconnected manifolds (the individual spheres). This thus also indicates how important the assumptions are that a dimensionality reduction method has to the underlying data distribution. If these assumptions are not met, the derived representation might not be meaningful, or at least not optimal.

2.6.3 Qualitative Evaluation on Image Data

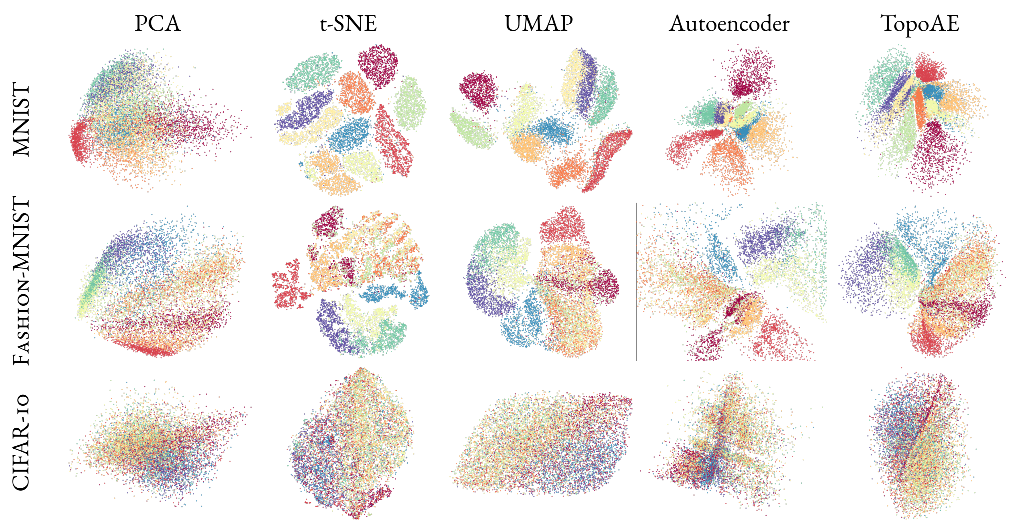


Figure 2.7: Visualization of latent representations derived from image datasets. Columns represent different dimensionality reduction methods, whereas rows correspond to different datasets. Colors correspond to classes in the datasets.

We compare different latent representations derived from image datasets by visualizing them in Figure 2.7. Here we can see interesting similarities between our method Topo-AE and UMAP. The similarities are particularly striking for the Fashion-MNIST and CIFAR-10 datasets, where both approaches give rise to a similar cluster structure and overall shape of the lower dimensional space. Even neighborhoods of classes are largely consistent when comparing the two approaches. Further, the individual classes represent relatively compact clusters in both cases. This is in line with the fact that both TopoAE and UMAP are topologically motivated [153].

This is not the case for the (non-regularized) autoencoder, which primarily pulls different classes apart in the latent space representation. Further, we can observe that t-SNE leads to clearly distinct clusters in MNIST and partially fragments clusters in the Fashion-MNIST dataset. On the MNIST dataset, both t-SNE and the autoencoder methods seem to lose some relationship information *between* clusters by pulling them apart. Interestingly, here the structure of TopoAE and the autoencoder are particularly similar. This could occur when the hyperparameter optimization procedure selects very small values for the regularization strength λ, which would reduce the impact of the topological loss term and thus make TopoAE closer to a standard autoencoder.

All approaches are having difficulties representing CIFAR-10 in a two-dimensional space. This is not surprising as it is the most challenging dataset of the ones considered and contains natural images which show much more variability compared to any of the other considered datasets. Nevertheless, our method seems to identify a linear substructure (in red) in the CIFAR-10, which is especially compact and separates the latent space representation into two parts. We can see a similar type of structure being present in the embeddings of UMAP yet here it is much less pronounced.

2.6.4 Quantitative Evaluation

The results of the quantification are provided in Table 2.2. Of all non-linear dimensionality reduction methods, TopoAE consistently preserves the multi-scale structure of the data quantified in terms of KL for different length scales. Here TopoAE is either the best approach or the second best behind PCA. The good performance of PCA is not entirely surprising as even a random linear projection would with high probability preserve distances in a lower dimensional space, according to the Johnson–Lindenstrauss lemma [114]. Further, TopoAE is competitive in terms of continuity, which indicates that neighbors in the data space are embedded close to each other in the latent space. In contrast, trustworthiness is usually lower for TopoAE which shows that far away points in data space are sometimes represented close to each other in the latent space. In this context, it is important to note, that it is only possible to reach both perfect trustworthiness and perfect continuity when points in data space reside in a linear subspace with the same dimensionality as the latent space. Only in this case would there exist an embedding that perfectly preserves all distances and thus neighborhood relationships.

Table 2.1: Evaluation metrics used for quantitative evaluation. Here n_i^K and ν_i^K denote the K-neighborhood of instance i where the former refers to the data space X and the latter to the latent space Z. Similarly, r_{ij} and ρ_{ij} denote the rank of instance j in terms of distance with respect to i for the two spaces.

Metric	Equation
KL_σ	$\mathrm{KL}\left(\mathrm{f}_\sigma{}^X \parallel \mathrm{f}_\sigma{}^Z\right)$
ℓ-MRRE [132]	$\frac{1}{N\sum_{k=1}^{K}\frac{\lvert N-2k\rvert}{k}}\sum_{i=1}^{N}\sum_{j\in n_i^K}\frac{\lvert\rho_{ij}-r_{ij}\rvert}{\rho_{ij}}$
ℓ-Cont [241]	$1-\frac{2\left(\sum_{i=1}^{N}\sum_{j\in\nu_i^K\setminus n_i^K} r_{ij}-K\right)}{N\min\{K(2N-3K-1),(N-K)(n-K-1)\}}$
ℓ-Trust [241]	$1-\frac{2\left(\sum_{i=1}^{N}\sum_{j\in n_i^K\setminus \nu_i^K} \rho_{ij}-K\right)}{N\min\{K(2N-3K-1),(N-K)(n-K-1)\}}$
ℓ-RMSE	$\sqrt{\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\mathrm{dist}(x_i,x_j)-\mathrm{dist}(z_i,z_j)\right)^2}$
Data MSE	$\frac{1}{N}\sum_{i=1}^{N}(x_i-\tilde{x}_i)^2$

Additionally, the topological loss only has a minor impact on the reconstruction error. This is important as it quantifies the degree of bias that we introduce with our approach. We expect the autoencoder to show better reconstruction performance as its mapping is not regularised by the additional loss term. If the reconstruction loss were very high, this would indicate that too much information about the input image is being lost due to the latent encoding.

The other quality metrics indicate the good performance of t-SNE on many datasets. Yet, these metrics are not directly comparable as it was necessary to evaluate the t-SNE performance on a subset of the training data due to a lack of a parametric mapping that could be applied to new data. Further, it is important to note, that many of these metrics focus on the conservation of local structure and do not respect how well the structure of data and latent space are aligned at different scales. This is further underlined by our experiments on the spheres dataset shown in Section 2.6.2. Here we see that t-SNE keeps the local structure in form of the spheres intact, yet distorts the overarching global structure of the larger surrounding sphere.

2.7 Conclusion and Discussion

In this chapter, we examined the problem of non-linear dimensionality reduction and presented a new approach that harnesses the expressive power of autoencoders and combines it with structure-preserving properties of topology. We showed that, under mild assumptions, we can propagate gradients through the computation of persistent homology and that the representations derived from persistent homology are also meaningful when computed on individual batches of a larger

Table 2.2: Quantitative evaluation of embedding quality according to multiple evaluation metrics. The hyperparameters of all methods were selected to minimize the objective $\mathrm{KL}_{0.1}$. For each metric and dataset, the best-performing model is highlighted **bold and underlined**, the second-best method is highlighted in **bold**. The column "Data MSE" shows the mean squared error of the reconstructed image compared to the original image.

Data set	Method	$\mathrm{KL}_{0.01}$	$\mathrm{KL}_{0.1}$	KL_{1}	ℓ-MRRE	ℓ-Cont	ℓ-Trust	ℓ-RMSE	Data MSE
Spheres	Isomap	0.181	**0.420**	**0.00881**	**0.246**	**0.790**	**0.676**	10.4	–
	PCA	0.332	0.651	0.01530	0.294	0.747	0.626	11.8	0.9610
	TSNE	**0.152**	0.527	0.01271	**0.217**	0.773	**0.679**	**8.1**	–
	UMAP	0.157	0.613	0.01658	0.250	0.752	0.635	**9.3**	–
	AE	0.566	0.746	0.01664	0.349	0.607	0.588	13.3	**0.8155**
	TopoAE	**0.085**	**0.326**	**0.00694**	0.272	**0.822**	0.658	13.5	**0.8681**
F-MNIST	PCA	**0.356**	**0.052**	**0.00069**	0.057	0.968	0.917	**9.1**	0.1844
	t-SNE	0.405	0.071	0.00198	**0.020**	0.967	**0.974**	41.3	–
	UMAP	0.424	0.065	0.00163	0.029	**0.981**	0.959	**13.7**	–
	AE	0.478	0.068	0.00125	**0.026**	0.968	**0.974**	20.7	**0.1020**
	TopoAE	**0.392**	**0.054**	**0.00100**	0.032	**0.980**	0.956	20.5	**0.1207**
MNIST	PCA	0.389	0.163	0.00160	0.166	0.901	0.745	**13.2**	0.2227
	t-SNE	**0.277**	**0.133**	0.00214	**0.040**	0.921	**0.946**	22.9	–
	UMAP	**0.321**	0.146	0.00234	**0.051**	**0.940**	**0.938**	**14.6**	–
	AE	0.620	0.155	**0.00156**	0.058	0.913	0.937	18.2	**0.1373**
	TopoAE	0.341	**0.110**	**0.00114**	0.056	**0.932**	0.928	19.6	**0.1388**
CIFAR-10	PCA	**0.591**	**0.020**	**0.00023**	0.119	**0.931**	0.821	**17.7**	0.1482
	t-SNE	0.627	0.030	0.00073	**0.103**	0.903	**0.863**	**25.6**	–
	UMAP	0.617	0.026	0.00050	0.127	0.920	0.817	33.6	–
	AE	0.668	0.035	0.00062	0.132	0.851	**0.864**	36.3	**0.1403**
	TopoAE	**0.556**	**0.019**	**0.00031**	**0.108**	**0.927**	0.845	37.9	**0.1398**

dataset. In the motivational example, we saw that TopoAE can capture nested relationships in a synthetically generated dataset of high dimensional spheres, whereas other methods fail to recover the multi-scale structure. This is particularly important as being able to cope with *several* manifolds is a challenging task in the field of manifold learning [31]. When evaluating TopoAE qualitatively on image datasets, we recognized its relation to other approaches inspired by topology such as UMAP, and observed that it is capable of generating highly structured latent spaces of high dimensional image data without discarding the relationships between clusters. In terms of evaluation metrics, TopoAE showed promising performance, especially when considering multiple evaluation scales. Further, the high performance in terms of continuity indicated that TopoAE is giving higher priority towards the preservation of points in the neighborhood of the data space while putting less weight on trustworthiness and thus allowing far-away data points in data space to be introduced into the latent space neighborhood.

While there exists a plethora of dimensionality reduction methods, our approach is the first to combine the expressive power of parametric non-linear dimensionality reduction via autoencoders with the ability of persistent homology to characterize manifolds [69]. Persistent homology has been used to evaluate dimensionality reduction methods in the past [196], yet its utilization in machine learning methods is still nascent [93]. We can relate this to some of the difficulties that arise when using the persistent homology framework: Most prominently, persistent homology operations are inherently discrete as creation and destruction events are also discrete. While our work shows that gradients can be propagated through the PH computation, the gradient does not contain any information about the PH computation procedure and instead assumes that it is constant (which it is for infinitesimal perturbations). A straightforward approach to how gradients could be estimated including the PH computation can be implemented using evolutionary strategies [204]. Here, noise vectors are sampled to perturb the input of the PH computation and the computation and decoding steps are repeated using different noise values. By multiplying the perturbation vectors with the resulting losses and computing the expectation over all noises, a gradient estimate can be computed for the input of the PH computation. Nevertheless, it remains unclear how this style of gradient estimation can be combined with back-propagation.

A further limitation of the proposed approach is the necessity of defining a distance metric on the input and latent representation. While this work uses the Euclidean distance metrics, it is arguably a bad metric for the computation of image distances and does not align with human perception. Here a promising line of research could be the utilization of deep neural network features as these highlight semantic and perceptual aspects of the data more prominently [270]. One of the issues with this approach is that features are influenced by the datasets the deep neural networks were trained on. This would not be problematic if instead *randomly initialized* neural networks were used for feature extraction, which has shown some success across domains [38, 229]. This is in line with follow-up work on TopoAE where alternative distance metrics lead to better separability of classes and improved visualizations [158].

Moreover, we limited the application of our approach to 0-dimensional topological features. While it is possible to extend topological autoencoders to higher dimensional topological features such as cycles and voids this could lead to problems when naively applied. Topological features are in general stable to many perturbations of the input data [54], yet the correspondence of a feature with the simplex that causes its creation or destruction is generally unstable [17]. Our topological loss term relies on the stability of this correspondence to derive meaningful gradients for the topological loss. Further, including high-dimensional features requires us to construct mappings for filtration values of higher-order simplices such as triangles. While this can, in theory, be easily implemented using the maximum of the simplices to ensure the required subset relationship of simplicial complexes, this would lead to sparse gradients that only influence single edges (i.e. the edges that either created or destroyed the cycle). Thus implementing full support for higher-order topological features, would require additional adaptations in the computation of persistent homology similar to Bendich, Bubenik, and Wagner [17]. Nevertheless, the exploration of such extensions to enable the use of higher dimensional topological features in topological autoencoders represents a promising direction for future research.

Furthermore, while our approach was inspired with biological data in mind, it has not yet been applied to it making further studies necessary. A first step could be an evaluation of the amount of data needed to derive meaningful representations. As autoencoders often require large amounts of data to be trained, this could make out approach unsuitable for some applications. Further, in order for the approach to derive meaningful results an appropriate distance metrics is required. In the context of single-cell data, this could be implemented using a metric that accounts for the probability of dropout reads in the sequencing data, or by preprocessing the data using an imputation strategy to remove those [238].

Finally, our approach relies on the predefined filtration of the Vietoris–Rips complex, which limits which aspects of the data are considered important for the persistent homology computation. While this is similar to the previous point, an interesting approach would be to *learn* the filtration during training. This was already demonstrated to be beneficial in the context of graph classification [97, 102]. In the context of our method, this of course leads to additional problems: If the filtration function can be learned, nothing would prevent the filtration from mapping all values to be very close to each other. This could be tackled by normalizing the filtration values and thus making the filtration scale invariant or by applying tricks from self-supervised learning. For example, filtration values of the input could be computed using an exponential moving average of the learned filtration function, which in other scenarios has been shown to mitigate collapse when applied appropriately [40].

Part II

Irregularly-sampled sequences

This part of the thesis concentrates on the development and evaluation of approaches for the classification of irregularly-sampled sequences often present in healthcare.

3 Uncertainty-Aware Antibiotic Resistance Prediction from MALDI-TOF Spectra

Matrix-assisted laser desorption ionization time-of-flight (MALDI-TOF) mass spectrometry (MS), is an important technique for the identification of microbes and has become a common tool in clinical routine [59]. The method involves embedding a to-be-analyzed probe in a matrix solution that stabilizes larger macromolecules. Then the components of the probe are fragmented and ionized using a laser and accelerated using an electric field. The accelerated particles are passed through a time-of-flight mass spectrometer, which essentially represents a long tube equipped with a sensor to detect ions at the opposing end. Due to the difference in mass and charge and thus speed of the individual particles, they can be distinguished according to their traveling time. More massive molecules and less strongly charged molecules will have a lower speed when entering the device[1] and less massive, more strongly-charged particles will be faster and thus traverse the device in a shorter amount of time. This allows the creation of a spectrum of intensities at different arrival times and thus different mass-to-charge (m/z) ratios.

While there are highly stochastic components to this process such as the fragmentation of larger molecules, the resulting spectrum is known to be characteristic of different microbes and has also been found useful in the prediction of bacterial characteristics, for example, the species of the microbe and its antimicrobial resistance [252].

Despite its high significance in the clinic, only very few machine learning approaches were proposed specifically for MALDI-TOF MS data. The following chapter takes this as a motivation and explores the construction of machine learning models that can directly be applied to the peaks of MALDI-TOF MS spectra. This is accomplished by the construction of a non-parametric preprocessing step, a novel heat-diffusion-inspired kernel for MALDI-TOF MS peaks, and the utilization of these in a Gaussian process classifier. While the method presented in the following sections was designed for and evaluated on MALDI-TOF MS spectra, it solely incorporates inductive biases for spectral data and not inductive biases specific to MALDI-TOF MS data. Thus the method can be

[1]The higher mass causes the particles to be accelerated more slowly in the electric field, whereas lower charge reduces the force applied to the particles in the electric field, thus also leading to less acceleration.

applied in a much wider setting than presented in this work and other types of mass spectrometry data represent well-suited candidates for its future application.

In the following, I will begin by providing some background on the methodology that our approach is based on by introducing kernels and Gaussian processes. I will then move toward the issues present in MALDI-TOF MS data and finally present our devised approach. This chapter is based on the following published work:
C. Weis*, M. Horn*, B. Rieck*, et al. "Topological and kernel-based microbial phenotype prediction from MALDI-TOF mass spectra". *Bioinformatics* 36, 2020, pp. i30–i38

3.1 Kernels

While it is evident that machine learning on MALDI-TOF MS spectra is important due to its potential clinical implications, the exact way how this would be conducted is largely unclear. MALDI-TOF MS spectra are irregular in the sense that they might contain a different number of peaks and these peaks do not need to be aligned. What is more, they are not represented in a way typically compatible with machine learning algorithms. A potential solution to this problem are *kernels*. Intuitively spoken, kernels represent similarity functions on objects and many machine learning methods can be expressed in terms of kernels [210]. They are thus a very lucrative approach when tackling machine learning problems on structured data as they solely require reasoning about the objects in terms of similarity and not in terms of the original data representation. A series of different kernels have thus been proposed for structured data ranging from kernels on graphs [24], kernels on irregularly sampled medical time series [136], and kernels on strings [144]. We introduce kernels formally using the definition below.

Definition 3.1 (Kernel). Given a non-empty set $\mathcal{X}$, a function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if there exists a Hilbert space $\mathcal{H}$ and a feature map $\phi\colon \mathcal{X} \to \mathcal{H}$ such that for all $x, x' \in \mathcal{X}$

$$k(x, x')\colon = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}. \tag{3.1}$$

A kernel thus requires the existence of a feature map ϕ to a Hilbert space $\mathcal{H}$ where the inner product of the features determines the value of the kernel. A Hilbert space is essentially a vector space equipped with an inner product, where the implied distances are complete for the space[2]. This of course gives rise to the question, of why we should use kernels at all if we need the existence of a feature representation in the end, why do not we directly use the feature representation to do

[2]Formally, this implies that all Cauchy sequences of elements in the Hilbert space are themselves elements of the space.

machine learning? Kernels have one additional advantage in this context: the feature representation can be very high dimensional or even *infinite dimensional*, as is for example the case for the RBF-kernel. Thus depending on the dimensionality of the kernel, it might be computationally very costly or even impossible to work with the explicit feature representation.

Interestingly, it is not always necessary to derive an implicit feature representation in order to ensure that a function is a valid kernel. The existence of a feature representation in some Hilbert space is intrinsically connected to a property of functions called *positive definiteness*. The Moore–Aronszajn theorem [6] states that given a symmetric positive definite function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ there exists a unique Hilbert space, in which the function is a *reproducing* kernel k and thus defined as the inner product of the space. The reproducing property reflects that the evaluation of any function $f \in \mathcal{H}$ can be represented in terms of an inner product with the reproducing kernel, i.e. $f(x) = \langle f(\cdot), k(\cdot, x)\rangle_{\mathcal{H}} = \langle f, \phi(x)\rangle_{\mathcal{H}}$, where $f(\cdot) = f$ represents the function coefficients and $k(\cdot, x) = \phi(x)$ is the feature representation of x.

3.2 Gaussian Processes

Gaussian Processes are a particular type of machine learning model, which allow regression (and due to extensions also classification) using kernels. Briefly put, a Gaussian Process (GP) is a stochastic process for which every finite collection of variables follows a multivariate Gaussian distribution. The subsequent introduction follows the book by Rasmussen and Williams [191]. In general, a GP can be seen to describe a distribution over functions f where $f\colon \mathcal{X} \to \mathcal{Y}$ and $\mathcal{X}$ represents the data domain and $\mathcal{Y}$ the prediction domain. A Gaussian Process can be completely specified by its mean function $m(x)$ and kernel function $k(x, x')$ which, based on an observed function f, are defined as:

$$\begin{aligned} m(x) &:= \mathbb{E}[f] \\ k(x, x') &:= \mathbb{E}\big[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))\big] \end{aligned} \tag{3.2}$$

We can then denote a Gaussian Process as $f \sim \mathcal{GP}(m(x), k(x, x'))$, which is equivalent to defining a prior over functions, where the kernel k captures how the function can vary over its domain. GPs are particularly attractive because their conditional distributions are themselves Gaussian distributions and can thus be computed in a closed form. Typically, we are interested in the *posterior distribution* of function values $\mathbf{f}_*$ at the location of the test points X_*, while *conditioning* on the training data X. For regression tasks, we want to compute the predictive distribution $\mathbf{f}_*|X_*, X, \mathbf{f}$

which, thanks to the properties of the Gaussian Process also follows a multivariate Gaussian/Normal distribution. This distribution of function values $\mathbf{f}_*$ at the points X_* can be expressed as

$$\begin{aligned}\mathbf{f}_*|X_*, X, \mathbf{f} \sim \mathcal{N}(&K(X_*, X)K(X, X)^{-1}\mathbf{f},\\ &K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)),\end{aligned}$$

where $K_{x,x_*} = k(x, x_*)$ represents the covariance matrix—evaluated using the kernel function—between the instances in the training set and the test set, respectively. While this can already be used to describe an idealized setting of observing the true function values without noise, observation noise is omnipresent. We can account for additional observation noise by describing the observed values as being drawn from the latent (unobserved) function with additive noise, such that $\mathbf{y} = \mathbf{f}(\mathbf{x}) + \epsilon$, where ϵ is drawn from a Gaussian distribution. This gives rise to conventional Gaussian process regression, where we optimize or sample the kernel and noise parameters according to the marginal likelihood of the model $p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)\,\mathrm{d}\mathbf{f}$, which can be computed analytically. Afterward, the predictive mean and predictive variance can be derived in closed form.

Extension to classification scenarios While Gaussian Processes can approximate functions, the Gaussian likelihood is often incompatible with downstream tasks. For example, when predicting a binary classification endpoint the image of the prediction function should be the range of probabilities $f\colon \mathcal{X} \to [0, 1]$. In contrast, the Gaussian Likelihood associates non-zero probability to values outside of this range and thus would also allow invalid values to be predicted. Luckily, we can adapt Gaussian Processes to all kinds of domains by utilizing tools of statistics for regression models. In particular, we can apply the idea of link functions used in generalized linear models [168, 191]. A link function typically transforms the response variable in a regression task to a continuous variable defined over the complete real line, and thus allows the application of linear regression to many different problem types.

In the context of Gaussian Processes we typically take the opposing perspective, we use the inverse link function, to transform our continuous random function into a variable that is "squished" to the correct function codomain. In this case, the latent function is called a *nuisance function* as we never actually observed its values but only its transformed representation. For example in a binary classification problem, the latent function f_* is transformed using the logistic function $\sigma(x) = \frac{1}{1+\exp(-x)}$ to represent probability estimates of the classes, resulting in a distribution of label predictions π_*. The prediction typically consists of two steps. First, the distribution of the latent function f_* at the test points $\mathbf{x}_*$ is computed conditional on the observed training data and labels via

$$p(f_*|X, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|X, \mathbf{x}_*, \mathbf{f})p(\mathbf{f}|X, \mathbf{y})\,\mathrm{d}\mathbf{f}, \tag{3.3}$$

where $p(\mathbf{f}|X, \mathbf{y}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|X)/p(\mathbf{y}|X)$ is the *posterior* over the latent functions. Afterward, predictions can be made based on the distribution of f_* by passing the function values through σ and computing the expectation of the label distribution, i.e. $p(y_* = 1|X, \mathbf{y}, \mathbf{x}_*) = \mathbb{E}[\pi_*] = \int \sigma(f_*)p(f_*|X, \mathbf{y}, \mathbf{x}_*)\,\mathrm{d}f_*$. Unfortunately, these integrals cannot be computed in a closed form and the posterior of $\mathbf{f}$ need not be Gaussian due to the nonlinearity of σ. It is thus necessary to apply approximation techniques to obtain a tractable solution.

A common approach is to apply the Laplace approximation to the posterior. The Laplace method approximates the posterior distribution of $\mathbf{f}$ in Equation 3.3 using a Gaussian distribution / second-order Taylor expansion at the maximum a posteriori estimate. The derivation of this approximation requires the computation of the Hessian of the log-likelihood $\log p(y|X)$. The Hessian is a matrix of all second-order partial derivatives of a function with respect to its input parameters. It is defined as

$$\mathbf{H}_f = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

We can use the Hessian to determine the covariance of a Gaussian which would have the same Hessian as our function by simply computing the inverse of the negative Hessian matrix, i.e. $\Sigma = \left(-\mathbf{H}_{\log p(y|X)}\right)^{-1}$. This provides us with an approximation to the posterior of the latent function values $\mathbf{f}$ denoted as $q(\mathbf{f}|X, y) = \mathcal{N}(\hat{\mathbf{f}}, \left(-\mathbf{H}_{\log p(y|X)}\right)^{-1})$ [149, 191]. Using the approximated posterior it is then possible to make predictions on new data by simply applying the predictive mean and variance equations of Gaussian processes to the approximated posterior of latent functions values as

$$\mathop{\mathbb{E}}_q[f_*|X, \mathbf{y}, \mathbf{x}_*] = \mathbf{k}(\mathbf{x}_*)^\top \nabla \log p(\mathbf{y}|\hat{f}) \tag{3.4}$$

$$\mathrm{Var}[f_*|X, \mathbf{y}, \mathbf{x}_*] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top \left(-\mathbf{H}_{\log p(\mathbf{y}|X)}\right)^{-1}\mathbf{k}_* \tag{3.5}$$

where $\nabla \log p(y|\hat{f})$ is the gradient of the log-likelihood given the most probable latent function values $\hat{f}$, $\mathbf{k}_*$ if the kernel is evaluated between all new data points $\mathbf{x}_*$ and the training data X. Given the mean and variance above, we can simply compute the expected prediction according to

$$\mathop{\mathbb{E}}_{q}[\pi_*|X, \mathbf{y}, \mathbf{x}_*] = \int \sigma(f_*) q(f_*|X, \mathbf{y}, x_*) \, \mathrm{d}f_*, \tag{3.6}$$

where $q(f_*|X, \mathbf{y}, \mathbf{x}_*)$ is the PDF of the Gaussian Distribution defined by Equation 3.4 and Equation 3.5. Further, when using the logistic function, the integral in Equation 3.4 can not be solved analytically and needs for example to be approximated via sampling or by using variational approximations [146].

As outlined above it is possible to derive an approximate posterior distribution for Gaussian Processes yet this does not yet allow for optimizing the hyperparameters of the kernel. Optimizing parameters of the prior is often referred to as *type-II maximum likelihood*, *empirical Bayes*, or *maximal marginal likelihood*. The marginal likelihood or model evidence is computed by marginalizing all latent variables. It is thus defined as

$$p(\mathbf{y}|\mathbf{x}, \theta) = \int p(\mathbf{y}|f) p(f|X, \theta) = \int \exp(\log p(y|X, f, \theta)) \, \mathrm{d}f \tag{3.7}$$

which by applying a Laplace approximation to the log-likelihood at $\hat{f}$ (the maximum a posteriori estimate) can be approximated as

$$\begin{aligned} q(\mathbf{y}|X, \theta) = & \exp(\log p(y|X, \hat{f}, \theta)) \\ & \int \exp\left(-\frac{1}{2}(f - \hat{f})^\top (-\nabla\nabla \log p(y|X, f, \theta)|_{f=\hat{f}})(f - \hat{f})\right) \mathrm{d}f, \end{aligned} \tag{3.8}$$

where the marginal likelihood is denoted as $q(\mathbf{y}|X, \theta)$ and $\nabla\nabla \log p(y|X, f, \theta)|_{f=\hat{f}}$ is the Hessian of the log-likelihood evaluated at the maximum a posteriori estimate. The statement above constitutes a Gaussian integral and thus can be solved in closed form to give rise to

$$\log q(\mathbf{y}|X) = \log p(y|X, \hat{f}, \theta) - \frac{1}{2} \det\left(-2\pi\nabla\nabla \log p(\mathbf{y}|X, f, \theta)|_{f=\hat{f}}\right) \tag{3.9}$$

which can be used to optimize the kernel hyperparameters θ. This is typically done by alternating iterations of gradient descent on the marginal log-likelihood above, and Newton optimization steps to derive $\hat{f}$ for the updated hyperparameters.

3.3 Issues with MALDI-TOF Spectra Processing

As MALDI-TOF data is a niche data source, specialized preprocessing pipelines have been developed by researchers [84] and companies [150]. These pipelines rely on a large number of hyperparameters, where common values are tailored with specific use cases in mind. This gives rise to difficulties when trying to transfer to new application domains, which have not previously been considered. The theoretical foundation of these concerns is rooted in the fundamental data-processing theorem [149] which is stated below:

Theorem 3.1 (Data Processing Inequality)**.** Assuming three random variables form a Markov chain $A \to B \to C$ such that their joint probability distribution can be written as $P(A, B, C) = P(A)P(B|A)P(C|B)$ then

$$\mathbb{I}(A; C) \leq \mathbb{I}(A; B), \tag{3.10}$$

where $\mathbb{I}(A; C)$ is the *mutual information* of the random variables A and C.

In simplified terms, it states that any type of information processing can only destroy information, as the *less processed* version of A, i.e. B, will always be more informative about A than its more processed variant C. A typical machine learning pipeline strives to reduce the amount of information between a representation and the input data, but increase mutual information between the representation and a label with additional processing.

This brings us back to our initial problem: complicated preprocessing objectives which involve many steps with decisions on hyperparameters will always have to be adapted when transferred to new applications. Due to the multi-step nature and the many involved hyperparameters in the processing pipelines, this can be a very challenging task. This is particularly important in the domain of MALDI-TOF data where spectra are often binned to create fixed-size representations which can be harnessed in off-the-shelf machine learning algorithms [58, 152, 242]. In general, we would expect that depending on the task the binning parameter might vary greatly. For example, a coarse binning might be sufficient when trying to derive a coarse label that can be easily determined by visually comparing spectra, yet for tasks that require more nuance in the examination of spectra this parameter might be completely off.

In our work, we show that instead of transforming the MALDI-TOF spectrum into a binned representation and thus losing a lot of the locational information of peaks we can yield improved performance by operating on the peaks directly. For this, we construct a conservative topology-based preprocessing pipeline that relies only on a single hyperparameter and devise a kernel that can operate directly on peak representations. I will introduce the two parts of our approach in the following.

3.4 Topology-Based Peak Detection

The peak detection algorithm we propose is based on the concept of persistence from computational topology [68] and constructs a filtration based on sublevel sets of a spectrum. A spectrum S is typically a set of n evenly spaced points mass to charge ratios m/z and associated intensity values, i.e. $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. In our method, we treat a spectrum as a function mapping from a compact domain $\mathbb{D} \subseteq \mathbb{R}$, i.e. the domain of x values defined in the spectrum, to the real values $f \colon \mathbb{D} \to \mathbb{R}$.

Further, we construct a filtration, i.e. a process which "grows" the function, from no points being present in the collection to all points being present in the collection, according to a predefined structure. In contrast to the other topology-inspired approaches in this thesis, we construct this filtration based on the *superlevel* sets of the function, i.e. $\mathcal{L}^+(c) := \{x \in \mathbb{D} \mid f(x) \geq c\}$. Two points (x, y) and (x', y') in $\mathcal{L}^+(c)$ are considered connected, if we can traverse from one to the other, without descending to a minimum which is lower than c. As a more creative example, imagine a hiker hiking in a flooded mountain range, where the level of the water corresponds to the value c. If we assume the hiker does not want to swim, he would then be considered connected (thus he can reach these points) to any mountain top, which does *not* require him to descend to a level lower than the water level c. As the level of the water decreases, he will be able to reach more and more points in the landscape until, when to water is completely gone, he can traverse the whole landscape.

If we apply persistent homology computations to this type of filtered function this leads to connected components being created at (global and local) maxima of the function and merged / destroyed at minima (as a minimum must always be in between two local or global maxima). These pairings are denoted in persistence diagrams as tuples of maxima and associated minima. It is important to note that in this context, we also account for maxima and minima at the boundaries of the superlevel set. In this simplified case we can look in more detail at how the persistence pairs are computed. We denote two points to be connected $x \sim_c x'$ given a level c if the path between them is a subset of $\mathcal{L}^+(c)$. The persistent homology computation maps each point x to the largest possible value c at which it would be able to reach a higher connected partner point x' within the superlevel set $\mathcal{L}^+(c)$. Thus any point will be mapped to a local minimum. We can define this pairing function $\pi : \mathbb{D} \to \mathbb{R}$ explicitly as

$$x \mapsto \sup\{c \leq f(x) \mid \exists x' \neq x : f(x') \geq f(x) \wedge x \sim_c x'\}. \tag{3.11}$$

π thus maps a point $x \in \mathbb{D}$ to a point that corresponds to the largest possible function value c where one can reach a higher point $f(x')$ within $\mathcal{L}^+(c)$. In the case of a global maximum, no such point exists (as there is no higher point that one could be connected to) and we define the supremum of the empty set to be the point which leads to the smallest function value $\sup \emptyset = \min_x f(x)$.

We can interpret this pairing as a *prominence* measure of a point. Points in between a maximum and a minimum will have $f(x) \approx f \circ \pi(x)$, as each point being added to the superlevel set will already be connected to points with larger function values. This means that the largest possible value c will correspond to the level set $\mathcal{L}^+(c)$ that x was just merged into such that in this case $c = x$. In contrast, the maxima will be associated with the minimum that connects them to an even larger maximum. Composing f and π we can construct a prominence map or persistence diagram $\mathcal{D}_f \colon \mathbb{D} \to \mathbb{R}^2$ according to

$$x \mapsto (f(x), f \circ \pi(x)) \tag{3.12}$$

and associate each value with a prominence $\operatorname{pers}(x) = |f(x) - f \circ \pi(x)|$ or the *persistence* of the topological feature in the persistence diagram. The encoding shown in Equation 3.12 is known to be stable to perturbations [54] such that two function which are close in terms of the Hausdorff distance will also be close in diagram space of Equation 3.12 in terms of the Bottleneck distance.

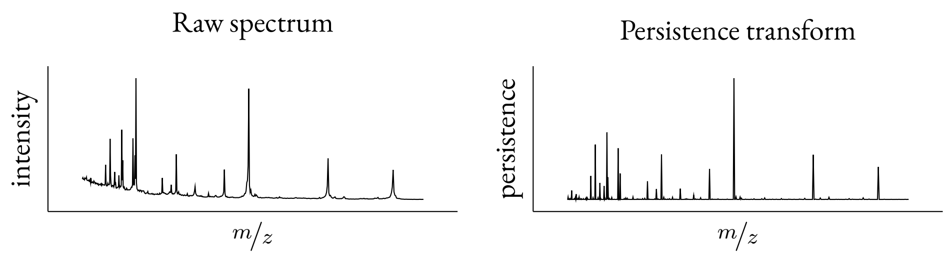


Figure 3.1: Example of a MALDI-TOF MS spectrum before (left) and after applying the persistence transform pre-processing (right). The pre-processing of the raw spectrum leads yields a simplified and cleaner representation of the spectrum which can be used in a downstream machine learning pipeline. The y-axis changes in interpretation as the values *do not* represent actual intensity values, but persistences of maxima. These are still of the same unit as the intensity and can be interpreted as a “relative” notion of intensity. From this representation we can easily create a sparse representation composed of the k peaks with the highest persistence.

Persistence Transform Finally, we describe how the computations above can be used to create an alternative representation of a MALDI-TOF MS spectrum, where many artifacts are removed. We derive this transformed spectrum representation $\tilde{f}(x) : \mathbb{D} \to \mathbb{R}$ by applying the persistence transform such that $\tilde{f}(x) := \operatorname{pers}(x)$. The persistence transform automatically detects peaks as local maxima are associated with large persistence values due to the construction of the superlevel set filtration. Further, it automatically removes the typical baseline intensity level present in MALDI-TOF MS spectra. We can see a visualization of the persistence transform applied to a MALDI-TOF MS spectrum in Figure 3.1. It is important to note, that this preprocessing does not perform any type of alignment of the peaks and does not depend on information from other spectra. It is a simple per-spectrum transform that allows us to derive a more tangible representation of a spectrum. We assume, that properties such as alignment or invariances to shifts of peaks should be implemented in the downstream machine learning model. This allows them to potentially be optimized using specific parameters and thus allows us (to some degree) to avoid issues due to the data processing inequality.

3.5 The Peak Information Kernel

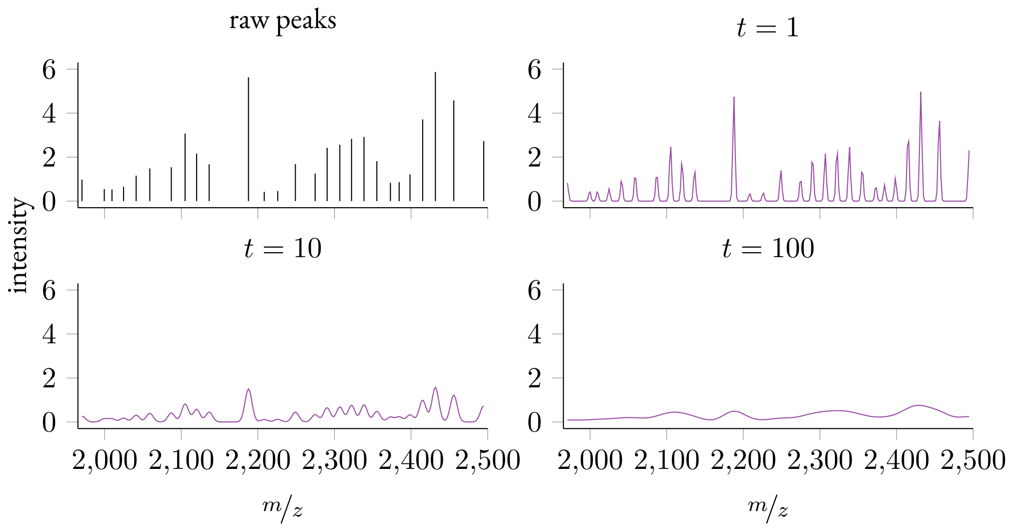


Figure 3.2: Visualization of the effect of the smoothing parameter t. Larger values of t lead to increased smoothing of the peaks. This means that peaks are more spread out and their exact position matters *less*.

One particular approach to work with these spectra is the construction of a data structure-specific kernel. This kernel should encode our understanding of the similarity between two spectra and allow us to leverage a large set of existing machine learning models on this type of data (see Section 3.1 for an introduction) [191, 210]. In the following, I will present the Peak Information Kernel (PIKE) designed for MALDI-TOF spectra, which is inspired by heat diffusion approaches for structured objects [16, 192]. We will start by explaining the feature representation of PIKE and will then show how to compute the inner product of the feature representation in practice.

Heat Diffusion Features The feature representation we propose here, is specifically constructed for sets of tuples, and thus does *not* require binning the spectrum into a fixed-length feature vector. This is achieved by operating on a smoothed functional representation of the individual peaks. Assume each spectrum is encoded as a set of tuples of the form $S = \{(x_1, \lambda_1), (x_2, \lambda_2), \ldots, (x_n, \lambda_n)\}$, where x_i represents the position of the peak on the m/z axis and $\lambda_i \in \mathbb{R}_{>0}$ is the intensity (or after using our topological preprocessing approach the *persistence*) of the peak. Further, define $\delta_{x_i}(x)$ to represent the Dirac delta function centered at a point $x_i \in \mathbb{R}$. We can then define the spectrum as the sum of these Dirac delta functions of defined height and position as

$$s(x) = \sum_{i=1}^{n} \lambda_i \delta_{x_i}(x), \tag{3.13}$$

where, $s(x)$ is 0 for any value besides the exact locations of the peaks and λ_i is the height of the i^{th} peak at the position x_i. We then model the features of each spectrum as a heat diffusion process that evolves starting from the boundary condition $s(x)$. Let $u(x, t)$ denote the solution to the following partial differential equation (PDE)

$$\frac{\partial u}{\partial t} = \nabla^2 u \tag{3.14}$$

$$\lim_{t \to 0} u(x, t) = s(x) = \sum_{i=1}^{n} \lambda_i \delta_{x_i}(x) \tag{3.15}$$

Assuming square-integrable functions on the real line, i.e. $u(x, t) \in L^2(\mathbb{R})$, this PDE has a unique solution. Nevertheless, as $s(x)$ is technically *not* square integrable, and the boundary should also be a solution to the PDE for $t = 0$, we define the boundary in terms of a limit. Practically, this means that we don't use $s(x)$ at the limit, but a twice integrable approximation to it. The closed-form solution to the PDE [199, Chapter 7] in this case is defined as

$$u(x, t) = \frac{1}{2\sqrt{\pi t}} \sum_i \lambda_i \exp\left(- \frac{(x - x_i)^2}{4t} \right). \tag{3.16}$$

From the perspective of kernel theory, we can interpret this as a parametrizable feature map $\Phi_t : \mathcal{S} \to L^2(\mathbb{R})$, i.e. mapping from the space of spectrums to the space of square-integrable functions. In this case, the functional representation itself represents the features associated with a spectrum and the resulting feature map is infinite-dimensional as this function is continuous. Given a spectrum S and $t \in \mathbb{R}$, we can denote the feature map by $\Phi_t(S) := u_S(x, t)$ where the subscript indicates that $u(x, t)$ is constructed using the peaks from S. Intuitively, we can interpret the feature map as a smoothed representation of the original spectrum. The larger the value t, the more *smoothing* we apply. For very large values of t the heat diffusion will dissipate all information of individual peaks until at some point the function would converge to a constant i.e.

$\lim_{t\to\infty} u(x,t) = c$. A visualization of functions generated for different values of t is shown in Figure 3.2.

Peak Information Kernel In the end we aim at using the feature map Φ to compute a *kernel*, i.e. a similarity between spectra. This kernel is typically computed using the inner product of the matching Hilbert space of the features. In our case, the feature space is the space of square-integrable functions, where the inner product is defined as the integral over the product of two functions, thus the kernel between two spectra S and S' is defined as

$$k_t(S,S') := \langle \Phi_t(S), \Phi_t(S') \rangle := \int_{\mathbb{R}} \Phi_t(S)(x)\Phi_t(S')(x)dx. \tag{3.17}$$

Given the structure of our functions, we can derive the above integral in closed form. While deriving the closed form we can drop some of the *error function* components which are equal to 1 given finite peak heights. This leads to the following equation for the computation of the kernel

$$k_t(S,S') = \frac{1}{2\sqrt{2\pi t}} \sum_{i,j} \lambda_i \lambda'_j \exp\left(-\frac{(x_i - x'_j)^2}{8t}\right), \tag{3.18}$$

where λ and λ' are peak heights from spectrum S and S' whereas x and x' denote corresponding positions of peaks. This represents a valid kernel, as a sum of squared exponentials with positive weights is still positive definite and thus a valid kernel [210]. Nevertheless, while being a valid kernel, Equation 3.18 shows some failure cases which could lead to unexpected results. In particular, if peak heights λ are less than 1, this would result in these being less similar compared to spectra with the same peak locations but peak intensities larger than 1. This is in contrast to our expectations that there should not be a threshold where peaks count more or less. We thus scale all peaks before the application of the kernel such that the smallest peak in the dataset has an intensity value of 1.

PIKE has a variety of properties, which we would like to highlight briefly. It is able to assess interactions between different peaks, as Equation 3.18 includes the distances between all pairs of peaks of the two spectra. Further, PIKE can easily be applied to spectra of different cardinalities without loss of information as would be the case when converting the spectra into a binning vector of fixed size. Finally, PIKE only contains a single continuous parameter t which governs the amount of smoothing applied to each spectrum. This has the particular benefit, that we can apply *type-II maximum likelihood estimation* (often also referred to as *empirical Bayes* or maximum marginal likelihood) [20] in order to derive values for the continuous kernel hyperparameter.

Of course, these properties also lead to limitations of PIKE. In particular, PIKE does not scale well to non-sparsified spectra, where very many peaks need to be compared due to its quadratic scaling in the number of peaks. While one can argue that this is not a limitation in practice as MALDI-TOF spectra typically consist of only a few hundred non-noisy peaks, it does prevent the application of PIKE to raw unprocessed spectra.

3.6 Experimental Evaluation

In the experiments we concentrated both on the evaluation of the classification performance of the model, and also on evaluating its confidence estimates. The latter is especially important in the clinical setting, where overconfident classifiers might (incorrectly) lead doctors to assume that a scenario is highly probable and act inappropriately.

Preprocessing We compare our approach to the common practice of applying logistic regression to the binned spectra. For the baseline, we preprocessed the spectra according to common practice, using the `MALDIquant` package [85] while relying on preprocessing parameters consistent with previous work [152]. The complete preprocessing pipeline is a multi-stage process which consists of (1) transforming spectra intensities, (2) smoothing, (3) baseline removal, (4) normalization, (5) peak detection, (6) peak frequency filtration, (7) warping of spectra and (8) trimming. This results in a total of 216 peaks selected by the pipeline which are then used to compute binned spectra for the input to the logistic regression model. For our topological preprocessing approach, we select the 200 most prominent peaks such that both approaches are comparable. For further details, on the exact preprocessing parameters we refer the reader to the original work [253].

Figure 3.3: Summary statistics of the dataset utilized in this study.

Species	Antibiotic	# samples	% resistant
E. coli	amoxicillin / clavulanic acid	1043	28.9
	ceftriaxone	1060	20.4
	ciprofloxacin	1051	29.7
K. pneumoniae	ceftriaxone	597	15.1
	ciprofloxacin	596	16.8
	piperacillin / tazobactam	576	13.9
S. aureus	amoxicillin / clavulanic acid	973	13.7
	ciprofloxacin	987	14.7
	penicillin	941	71.4

Experimental Setup In our experiments, we concentrate on the task of microbial resistance prediction as it is of high clinical relevance. The dataset used in this study was acquired at the University Hospital Basel and consists of 2676 MALDI-TOF MS spectra collected for clinical routine measurements in the year 2008. It is composed of measurements from three different species of bacteria *Escherichia coli* (*E. coli*, 1068 spectra), *Krebsiella pneumoniae* (*K. pneumoniae*, 603 spectra), and *Staphylococcus aureus* (*S. aureus*, 1005 spectra). For each species, we utilize the resistance information on antibiotics typically considered in treating the respective species. A summary of the available prediction endpoints for each species and the prevalences can be found in Table 3.3. Each species–antibiotic combination is treated as a binary classification problem. For further detail on the acquisition of the spectra, we refer the reader to the original publication of this work [253].

All classifiers are evaluated on 5 random class-stratified splits, where 80% of the data where selected for training and 20% for testing, report mean and standard deviation of the test split performance metrics. In order to select the hyperparameters for the logistic regression model, each individual train split is used in a 5-fold cross-validation scheme. We select the hyperparameters for each random split such that they have the highest average performance on the cross-validation test splits and then refit the model to the complete training data of the random split. For the Gaussian Process classifier in combination with PIKE, we instead rely on type-II maximum likelihood to optimize the kernel hyperparameters on each random training split. To simulate a balanced setting, we oversample the minority class for all approaches during training but do not apply oversampling during testing. This ensures that the model does not account for the prevalence of classes when predicting. Otherwise, the model might treat errors in the minority class as less important due to the lower probability of them occurring.

For the logistic regression baseline, we bin spectra into same-size bins, where the intensity values of each bin are formed by the sum of the intensities of all peaks which get allocated to the bin. We select hyperparameters from a large grid of possible combinations. In particular, we vary the number of bins to bin the spectrum (300, 600, 1800 and 3600), the regularization (L_1, L_2, $L_1 + L_2$ and no regularization) and the regularization strength (10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3} and 10^{4}).

3.6.1 Antibiotic Resistance Prediction

Summary of Results We present the results of the study in Table 3.1, where to increase readability we always only refer to the first antibiotic of an endpoint if multiple antibiotics are grouped as shown in Table 3.3. We compare the application of logistic regression (LR) and a Gaussian Processes (GP) with an RBF kernel on the `MALDIquant` preprocessed and binned spectra in order to disentangle the effects of preprocessing, kernel, and downstream classifier. We find that when as-

Table 3.1: Results for antibiotic resistance prediction. The performance of the methods on the individual binary classification tasks is shown in terms of the mean area under the precision-recall curve (AUPRC). Each method is composed of two parts, the preprocessing part, which shows if the `MALDIquant` preprocessing pipeline (MQ) or the persistence transform pipeline (PT) was used, and the downstream model, which can be one of logistic regression with binning (LR), Gaussian Process with binning and RBF kernel (GP-RBF) and Gaussian Process with PIKE (GP-PIKE).

Species	Antibiotic	MQ–LR	PT–LR	MQ–GP–RBF	PT–GP–PIKE	MQ–GP–PIKE
E. coli	amoxicillin	40.96 ± 7.41	35.72 ± 2.70	32.50 ± 8.48	38.89 ± 2.03	**47.07 ± 3.85**
	ceftriaxone	63.22 ± 6.08	58.04 ± 3.14	46.29 ± 24.00	62.78 ± 3.19	**70.64 ± 3.21**
	ciprofloxacin	61.37 ± 8.52	55.14 ± 3.84	34.65 ± 10.71	54.02 ± 4.04	**67.99 ± 3.01**
K. pneumoniae	ceftriaxone	58.20 ± 9.79	56.47 ± 6.26	58.72 ± 25.29	72.38 ± 9.03	**77.04 ± 6.82**
	ciprofloxacin	41.71 ± 9.82	35.04 ± 7.74	30.88 ± 13.54	40.15 ± 13.29	**54.63 ± 10.12**
	piperacillin	31.58 ± 6.81	38.62 ± 8.65	13.79 ± 0.00	48.95 ± 9.90	**56.46 ± 9.68**
S. aureus	amoxicillin	52.88 ± 3.91	55.21 ± 4.08	13.85 ± 0.00	61.02 ± 12.45	**69.15 ± 9.15**
	ciprofloxacin	34.11 ± 3.26	26.30 ± 6.16	23.32 ± 11.88	30.51 ± 2.95	**39.37 ± 6.62**
	penicillin	79.66 ± 3.34	79.61 ± 4.66	74.15 ± 3.15	80.67 ± 1.92	**83.17 ± 3.54**

suming conventional preprocessing, the LR model always outperforms the GP model when using an RBF kernel, yet the GP combined with our kernel PIKE performs *consistently* better than all comparison approaches and in many cases outperforms the LR classifier by a large margin. Interestingly, we find that `MALDIquant` preprocessing is very important for good performance and the persistence based preprocessing cannot to a satisfying degree replace the established multi-stage preprocessing pipeline. Generally, we observe that the different experiments lead to very strong differences in improvement over the baseline. This could for example be explained by different resistance mechanisms which would manifest themselves to different extents in the MALDI-TOF MS spectrum.

Performance of MQ–GP–PIKE Method Our proposed kernel PIKE combined with a Gaussian Process classifier applied to the `MALDIquant` processed spectra leads to superior performance in all tested scenarios. We can attribute this to two factors. First, PIKE is capable of considering non-linear interactions between peaks due to the smoothing process. This is especially important as proteins might be fragmented into multiple smaller ions and thus the presence of a protein in the cell could be linked to several peaks in the spectrum. Further, each ion can be charged with multiples of the elemental charge of an electron, such that the same fragment could also occur at multiple discrete locations in the spectrum. While a binned spectrum can represent the multiple versions of a protein, it aggregates the presence of peaks over an interval, such that these cannot be differentiated afterward. An additional benefit of PIKE stems from the fact that it only relies on continuous parameters, which allows for type-II maximum likelihood estimation when combined with a GP. In contrast, the number of bins cannot be optimized as part of the model training and thus needs to be evaluated over a grid of potential values lowering the resolution of the search and potentially missing the optimal value. Finally, PIKE was designed such that minor misalignments between peaks can be compensated by increased smoothing. This allows the selection of a t parameter by the GP which would compensate optimally for the amount of "mismatch" in peak positions.

Effect of GP Kernel As shown in the results, the choice of data representation and kernel is very important for good performance of the GP. The approach of utilizing the binned spectra in combination with an RBF kernel leads to significantly worse performance compared to applying PIKE to the peaks of the spectrum directly. Given the featurization of binned spectra, logistic regression typically performs favorably compared to the GP with RBF kernel, which indicates that the higher performance of the GP-PIKE approach is mostly attributable to the PIKE kernel and its direct operation on the peaks of the spectra. While it is also possible to apply a kernel in kernel regression or support vector machines we concentrate on utilizing Gaussian Processes in this study due to their probabilistic nature and prior reversion when provided with samples far away from the training data. We will explore this aspect of GPs in more detail in the next section.

Effect of Preprocessing When comparing the performance of the topological preprocessing proposed in this work (PT–LR) with the conventional preprocessing pipeline (MQ–LR), we observe that the conventional pipeline usually leads to better performance in terms of the area under the precision-recall curve (AUPRC). Nevertheless, in many scenarios, the differences in the approaches are not significant. We hypothesize that these differences are mainly due to the warping step, which aligns the peak positions of the spectra based on prominent peaks. In contrast, our preprocessing pipeline does *not* change the location of peaks, such that shifts along the m/z-axis cannot be corrected. This can to some degree be accounted for in the kernel by selecting higher smoothing values t, which explains the improved performance of PT-GP-PIKE compared to PT-LR. The gains of using aligned peaks seem to be much larger though, as the smoothing also leads to a loss of information as peaks become more strongly spread out and merge into other peaks. It is important to mention though, that warping is not always possible especially if the dataset contains very heterogeneous samples. The warping step requires the existence of reliably measured peaks such that they can be used as landmarks for the alignment of the spectra. What is more, the warping step needs to utilize all data, which is contrary to the common practice of machine learning, where performance is evaluated on a strictly held-out part of the data to avoid overconfident generalization estimates. We thus conclude that while the topological preprocessing might not be beneficial in our specific setup, there are scenarios in which it might be a good choice. In particular, we conjecture that the development of more advanced kernels for MALDI-

TOF spectra (which might account for shifts in the m/z position without loss of information) could lead to significant improvements in this approach.

3.6.2 Rejection of Low-Confidence Samples

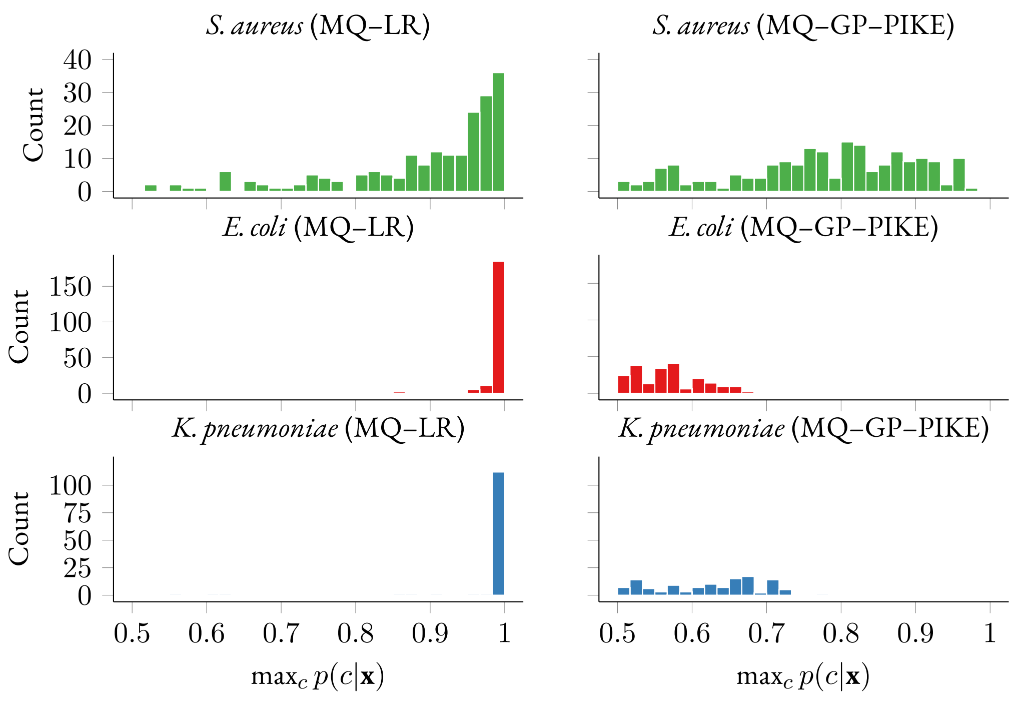


Figure 3.4: Histogram showing the different distributions of the maximum class probability $\max_c p(c|\mathbf{x})$ for the logistic regression (left column) and the Gaussian Process classifier with PIKE (right column) trained on *S. aureus*. The upper figure depicts the in-training distribution of maximum class probabilities, i.e. class probabilities with respect to *S. aureus*, while the middle and lower figures show the values for out-of-distributions species (*E. coli* and *K. pneumoniae*). It is visible that the distribution of confidences for the logistic regression model is highly skewed to high probabilities even on data the model was not trained on. In contrast, the Gaussian process model shifts probability mass to lower probabilities when applied to the out-of-distribution data, and thus recognizes appropriately that no clear prediction can be made in this case.

In a clinical setting, it is important that a model can also signal to the clinical practitioner when no confident prediction can be made. This for example could be the case when samples are very close to the decision boundary of the model, which is commonly referred to as aleatoric uncertainty. Further, this could also occur when data provided to the model is *very different* from the training data, which is referred to as epistemic uncertainty. In both cases we would want our model to ideally be able a *reject* a sample such that the clinician can rely on other data for making a decision. While most classifiers can in theory reject samples that are close to the decision boundary, only a few are able to understand if a given data point is close to their training distribution. One example of a model which can recognize the latter case is a Gaussian Process. Combined with an appropriate kernel, Gaussian processes show the property of prior reversion, such that predictions far away from the training data become increasingly close to the prior [142].

In the context of predictions on MALDI-TOF MS spectra, this is extremely relevant and important. Here a model might be confronted with spectra from species that have not been seen during training when for example a strain was picked up during travels in a distant country. In these cases, it is of crucial importance for a doctor to purely rely on the antibiotics resistance tests and not on the predictions of the machine learning model. To investigate if our proposed approach abides with the above requirements, we run a series of additional experiments. In particular, we extract the confidence of the classifier on the data points by noting the maximum class probability over the classes $\max_c p(c|\mathbf{x})$. In an ideal scenario, we would want the classifier to have high confidence on *all* samples drawn from the same distribution as the training data and low confidence on out of distribution samples. This would allow the application of a threshold $\theta \in [0.0, 1.0]$ that only keeps samples satisfying $\max_c p(c|\mathbf{x}) > \theta$ and thus reject out-of-distribution samples.

We can see a comparison of the distribution of confidences for both MQ–LR and MQ–GP–PIKE on in-distribution (ID) and out-of-distribution (OOD) data in Figure 3.4. While both approaches show a relatively wide distribution of confidences for the in-distribution data of the species *S. aureus*, the predictions of MQ–LR are skewed toward higher confidence values. The skew does not resemble a problem in the training data, yet the problem of overconfidence on OOD becomes evident when examining the other confidence distributions. Here MQ–LR predicts *higher* confidence than ID data due to its inability to estimate epistemic uncertainty. In contrast, the MQ–GP–PIKE approach clearly assigns lower confidence values to the OOD data, which would allow the construction of a thresholding rule as previously described. The GP is able to recognize that these samples are not close to the training data and thus with increasing distance from any training sample, reverts to the prior probability of 50%. The ability to recognize OOD data of course also relates to the feature representation: While the probability that all peaks are distributed the same way for different datasets is relatively low (as we have exact peak locations), this is not the case if we apply binning. It is important to note that while we expect the GP to be able to well recognize OOD data for small values of t, this does not have to be the case for larger values of t. We can attribute this to the fact that for large values of t, our kernel PIKE does not have to converge to low similarities as the spectra become increasingly distinct.

To characterize the degree to which OOD samples can be rejected without rejecting ID samples we compare the rejection ratios of OOD and ID samples over different thresholds and different splits in the left panel of Figure 3.5. It is evident that using our approach we can reject most OOD spectra while only rejecting very few ID spectra and that this holds if we compare different types

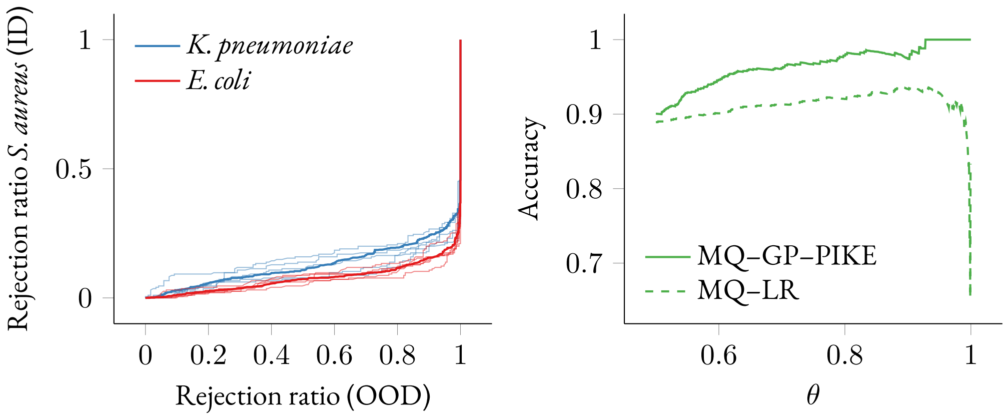


Figure 3.5: Left: Trade-off between the proportion of rejected in-distribution and out-of-distribution samples of MQ–GP–PIKE trained on *S. aureus* for amoxicillin resistance prediction. Right: Accuracy when rejecting samples according to threshold θ for MQ–GP–PIKE and MQ–LR trained on *S. aureus* for amoxicillin resistance prediction.

of OOD spectra. With our method, it is possible to reject all out-of-distribution samples, while only rejecting 30% of the in-distribution data. This very conservative threshold could be a potentially suitable trade-off in clinical contexts in order to avoid false positive predictions on which the method would not be considered reliable. The ideal threshold is of course highly dependent on the risks and costs of rejecting ID vs OOD samples and should be determined by a clinical practitioner.

Interestingly, compared to the MQ–LR approach our method can better represent the actual uncertainty of the prediction, as accuracy improves while rejecting more uncertain samples as is shown in the right panel of Figure 3.5. While the MQ–LR approach also shows slightly improved performance when rejecting samples close to the decision boundary, the rate of improvement is much lower and the performance deteriorates strongly when reaching the highly confident regime. This highlights that the MQ–LR model assigns high confidences to samples that it classifies incorrectly.

3.7 Conclusion and Discussion

In this chapter, I presented MALDI-TOF MS data and highlighted how current approaches for machine learning in this domain suffer from issues that hinder reproducibility and transfer of models between different hospitals. I presented a novel approach for peak extraction from MALDI-TOF MS data based on notions from persistent homology. In our work, we showcase that it relies on a single parameter and partially removes the necessity for complicated multi-stage preprocessing pipelines. Further, we show that our heat-diffusion-inspired Peak Information Kernel (PIKE) avoids the necessity of binning the spectra to derive fixed-size representations used in the field to train conventional machine learning algorithms. PIKE only contains a single continuous parameter, making it amenable to optimization using type-II maximum likelihood and allowing it to be combined with probabilistic Gaussian Process classifiers. We showcase how our kernel can be used in a clinical antibiotic resistance prediction scenario and highlight that our approach outperforms conventional machine learning pipelines on MALDI-TOF MS spectra. Further, the quantification of uncertainty allows our approach to recognize spectra that are out-of-distribution with respect to the training data and to signal clinicians that predictions in this realm are unreliable due to the lack of data coverage. We demonstrate the importance of this property in detailed comparisons to a logistic regression baseline which shows strong overconfidence on out-of-distribution data and would thus lead to false predictions with potentially negative downstream effects in the clinic.

MALDI-TOF MS data is used in a variety of clinical diagnostics and many hospitals rely on commercial products to obtain information about what is contained in a sample [59, 252]. Most machine learning approaches for MALDI-TOF MS data rely on a two-stage approach where features are extracted from the preprocessed spectra, (typically via binning) and the vector representations are used in conventional machine learning models. This has the disadvantage that much of the fine-scale information of MALDI-TOF MS spectra is lost as peaks get aggregated. While this can also be beneficial to make features invariant to lab preparation and machine settings, the locations and sizes of bins are typically chosen arbitrarily with little regard for the data which dampens potential benefits. Our work shows that the design of methods specifically for MALDI-TOF data can lead to highly-promising results. It indicates that specifically in MALDI-TOF data, there could still be a large room for improvement as many properties of the data-generating process can be incorporated into machine learning models. One such direction would be to incorporate invariances concerning the m/z ratio. After the ionization process, a fragment with mass m can have a charge z that is an arbitrary multiple of the electron mass e. This process is to some degree random such that an ideal machine learning algorithm should be able to exploit invariances concerning the difference in charge multiples. For example, a fragment in a sample with $z_1 = 3e$ charge should be considered similar to a fragment with the same mass but $z_2 = 2e$ charge. The inclusion of such invariances or inductive biases is a very promising direction for new machine learning methods that significantly improve performance and robustness to distribution shifts.

4 Classification Models and Evaluation Strategies for Medical Time Series

Healthcare is one of the sectors where ML is posed to have the largest impact on improving human lives and the standard of care and living [55]. One particular domain where machine learning algorithms can support clinicians is the monitoring of clinically ill patients as these cannot be continuously monitored by doctors due to the diverse set of tasks they are responsible for [70]. This can lead to information being missed or not reacted upon on time which increases the chances of mortality for ICU patients [65]. While many studies have indicated links between mortality and insufficient nurse staffing [72] the statistical significance of these associations is *not* consistently deemed significant [173].

Due to the large success of recent machine learning methods in the analysis of complicated signals ranging from the classification of images to the processing of human language [129], these approaches represent promising candidates for pushing the frontier of clinical care. Here in particular clinical decision support and monitoring systems can support doctors by providing additional information or by monitoring patient state around the clock to ensure speedy responses [231]. This chapter will focus on monitoring systems and will two aspects of medical time series data that are especially critical: data compatibility and model transferability. We will start by characterizing the issues present and then dive into potential solutions.

This chapter is partially based on the following published works:

- M. Horn et al. "Set functions for time series". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 4353–4363
- M. Moor*, N. Bennet*, D. Plecko*, M. Horn*, et al. "Predicting sepsis in multi-site, multinational intensive care cohorts using deep learning". Preprint. 2021. arXiv: `2107.05230 [cs.LG]`

4.1 Issues of Medical Time Series Series for Machine Learning

In the following, I will present common issues that arise when trying to apply machine learning to medical time series data. These issues concentrate on the properties of the data and do not consider for example biases due to sampling, which represents a further important issue regarding the application of machine learning models to the medical domain [246]. These issues are not discussed here as they are not generally addressed by the algorithms and methods presented in this thesis.

4.1.1 Irregular Sampling

Medical time series as they predominantly occur in the Intensive Care Unit (ICU), have many unique characteristics compared to other time series on which machine learning models are typically applied. The most striking difference is that medical time series are irregularly-sampled, such that not every measurement is available at every time point. This has to do with the way these time series are acquired. While some measurements of patient vitals are continuously monitored using bedside machinery (such as heart rate, respiratory rate, and blood pressure), others require interventions from nurses or doctors. The latter is the case for lab measurements which require for example sampling of the patient's blood and are often only done when explicitly requested by a doctor and responsiveness scores which require nurses to perform a series of check-ups with the patients.

As it is impossible in the highly dynamic day-to-day work in the clinic to guarantee that measurements are done at perfectly regular intervals and as the acquisition of lab measurements induces additional costs which should not be done without the necessity of the results, the measurements of medical time series do not fall onto a fixed spacing grid. This makes it problematic for modern machine learning methods to be directly applied to this type of data. While classical machine learning approaches which include feature engineering pipelines can avoid these issues by using summary statistics over multiple horizons and imputation of missing values [108, 162], modern gradient-driven representation learning approaches need to resort to alternative solutions as they try to derive the appropriate feature representations for the data as part of the learning process.

Bridging gradient-based learning and irregular sampling The most straightforward approach towards applying models in this domain is to simply treat the issue as a missing value problem. In a Bayesian framework, missing data would be treated as a latent variable over which we can perform inference to derive a posterior distribution over values that the unmeasured variable could potentially have. Then predictions are made by marginalizing this unobserved variable, i.e. by considering all possible values of the unobserved value and weighting the predictions according to the probability associated with the value under consideration. A direct application of this approach is unfortunately not feasible in most cases, as an irregularly sampled time series also does not attain to any grid which would allow stating that at this particular time point, certain measurements are missing. It is thus common to rely on binning of observations into discrete intervals to construct this regular grid representation [12]. Nevertheless, even in this context, the use of naive Bayesian inference is intractable for deep neural networks and it is necessary to apply approximation methods such as variational inference [166].

An alternative approach is to impute the data and augment it with indicators that provide the model with the information if a value was actually measured or if the value that is presented to it is imputed [141]. This strategy was further extended by allowing the model to decide how imputation should take place. Che et al. [47] suggest allowing a Recurrent Neural Network (RNN) to learn feature-specific decay rates which model how fast the value in question decays to the empirical mean of the feature if it is not observed. This enables the model to derive a feature-specific notion of time and thus to represent the fact that some features will probably not change as fast as others. While these approaches are applicable to irregularly-sampled data, they either rely on imputation schemes or empirical global estimates of the data distribution, without taking correlations between channels into account.

The previously described approaches give models some flexibility regarding the interpretation of non-measured values, yet imputation approaches are often highly constrained and rely on very simple models for the underlying generative process. Despite it certainly being possible to apply advanced time series imputation methods such as Gaussian Processes to model the structure present in the data in more detail, these give rise to further problems: The imputation might actually lead to loss of information needed for the downstream task due to the data processing inequality. To tackle the loss of information many recent approaches have therefore concentrated on learning the imputation strategy which generates a fully observed regularly sampled time series together with a downstream (classification) model in an end-to-end fashion [80, 137, 159, 219]. The motivation behind this is to ensure that the parameters used for imputation are in line with the downstream objective and thus reduce the amount of information that could be lost due to the imputation procedure. This is achieved by stacking the imputation approach (typically Gaussian Processes, Multi-Task Gaussian Processes [23], or kernel-based interpolations [219]) with a downstream classifier and optimizing the parameters of the imputation via back-propagated gradients of the classifier. Typical choices for downstream classifiers include variants of recurrent neural networks such as LSTMs [80] or temporal convolution networks [159] — due to the imputation module any downstream classifier that operates on fully observed regularly sampled time series can be applied.

Learning Imputations is Hard Most of the aforementioned approaches have in common, that they construct a fully observed time series which is then used in a downstream model. Given that data is only sparsely available, this constitutes a difficult task. Taking the idealized perspective, the best thing a model could do to solve the imputation problem is to first learn a model of the underlying dynamics of the system which adequately describes correlations that occur in the time series features over different time scales. Then the model should derive the specific instantiation parameters of the time series that is currently being observed, i.e. perform inference with respect to this model. Finally, the model should simulate the evolution of the underlying dynamical system and provide estimates of the distribution of values that would occur at different time points where no data is observed. If we knew that we had access to the true underlying model which gave rise to the data, i.e. the true causal process, inference with respect to this process would be our best shot at deriving realistic imputations which transfer to different tasks and different settings [208, 248].

Nevertheless, this underlying model is typically not known and cannot be derived reliably from data. Thus it raises the question if it is desired to work with models which can only approximate the true imputations solely for the purpose of being able to tackle a discriminative learning problem. Generally, as we argue in later parts of this section, it may instead be more favorable to construct models which can directly be applied to medical time series data.

4.1.2 Distribution Shifts Between Hospitals and Countries

A further issue when applying machine learning models in the clinic is the presence of distribution shifts. Studies have shown that even the admission rates for diseases and accidents vary between countries after accounting for differences in demographics [175]. These differences are additionally amplified by the variability in clinical practice in the same country as even academic medical centers do not provide consistent treatments in cases where clear clinical practices exist [255]. Additionally, medicine is a highly complicated field in which the opinions of experts may often diverge, such that it is not always possible to objectively prefer one treatment approach over the other. In such cases, patients are given the freedom to decide over the treatment which often results in them mirroring the opinion of their treating doctor [111].

These differences in admission rates and clinical practices lead to subtle distribution shifts in the data. For example, if doctors in one hospital always administer vasodilators (medication to reduce high blood pressure) when the blood pressure reaches a certain level and another hospital defines this threshold differently or simply does not prescribe vasodilators as frequently, this would lead to a distribution shift that could even be recognized in the marginals of the distribution. While there are some indications that this is an issue for recently developed models [14, 108, 258], no general large-scale study which examines the transferability of machine learning models between hospitals has yet been performed. A first step towards improving the current state is the definition of gold standards for the evaluation and monitoring of deployed algorithms which has recently been an ongoing topic of discussion in the community [246].

4.1.3 Circularity in Label Definitions

A recent study by Schamoni et al. [207] highlighted the problem of circularity in the context of sepsis definitions. In the case of the sepsis onset prediction task, it is especially difficult to obtain time-resolved labels, as hospital billing codes are highly unreliable for endpoint definitions [28, 77] and do not contain any information on the exact time at which sepsis occurs. It is thus common to construct labels using an algorithmic definition of sepsis onset derived from clinical guidelines, the applicability of which of course implies that these data are available and readily measured. In many cases, the same data is then used to train a machine learning algorithm, which will trivially be able to achieve high performance as it solely needs to mirror a simple algorithmic definition [207].

This gives rise to two main problems. First, as the labels do not represent an independent ground truth, the approaches are compromised and cannot actually be validated/tested on truly independent data. Second, the knowledge derived from such approaches will be highly limited in its scope as a supervised machine learning algorithm will mostly try to learn the structure of the label definition. This has been observed in some studies that try to bridge the gap between machine learning and the clinic [108, 162].

4.1.4 Indication Bias

A further issue can arise if models do not make independent decisions, but decisions based on the doctors' actions. Many treatments can give the model information about the doctors' expectations regarding the patient. For example, the administration of antibiotics can tell the model that the medical practitioner suspects that the patient is infected or expects an infection to occur. What is more, even the fact that a lab test for infection was requested would be sufficient to convey this information. In randomized trails, this is often referred to as indication bias or confounding by indication [75] and captures the bias introduced when a treatment is incorrectly associated with an outcome as both are causally linked to the same underlying indication or cause. For example, if a patient has a fever and is given an antipyretic, i.e. a medication to reduce fever. An observational study might show that the patients that where treated with an antipyretic show higher rates of mortality, yet the association might be spurious as the true cause for the higher mortality might as well be the fever measured by the doctor.

In the context of machine learning, this occurs when the model (incorrectly) attributes the deterioration of the patient's state to the actions/treatments of the doctor. While this might actually result in a good predictor the derived models are unlikely to generalize due to differences in treatment routines as mentioned in Section 4.1.2. Additionally, the deployment of such systems brings even higher risks such as feedback loops due to the interactions between machine learning models and medical practitioners [246].

For example, consider the scenario of a diagnostic support system that predicts the deterioration of patients. A common cause of patient deterioration and death is a systemic infection or sepsis. If the model has access to lab measurements or treatments of the doctor, it will most likely rely on the doctor's treatment variables or the request of the doctor to do lab measurements in order to predict infection. Issues may arise after the model has been deployed in a hospital for a prolonged time period and the model's predictions seem in line with the conclusions the doctors are making independently of the model. As confidence in the system rises, the model's predictions might start to get included in the decision-making process of medical practitioners such that the measurements of antibiotics are not made anymore purely based on suspicion by the doctor but also on the risk score of the model. As the model relies on the confounding signal of antibiotics measurements from the doctor this would lead to a feedback loop that could deteriorate the state of clinical care if not adequately monitored.

It is important to note that many current studies do not account for this type of circularity and do not consider it problematic that the model has access to sampling information [108, 124].

4.2 Tackling Irregular Sampling – Encoding Time Series as Sets

Motivated by the issues of incompatibility between modern neural networks and irregularly sampled time series described in Section 4.1.1, we propose an alternative perspective in order to train models on irregularly sampled time series without prior imputation. Our method is driven by the understanding that, while RNNs and similar architectures are well suited for capturing and modeling the dynamics of a time series and thus excel at tasks such as forecasting, retaining the order of an input sequence can even be a disadvantage in some scenarios [245]. We show that by relaxing the condition that a sequence must be processed in order, we can naturally derive an architecture that *directly* accounts for (i) irregular sampling, and (ii) unsynchronized measurements. Our method SeFT: Set Functions for Time Series, extends recent advances in set function learning to irregular sampled time series classification tasks, yields favorable classification perfor-

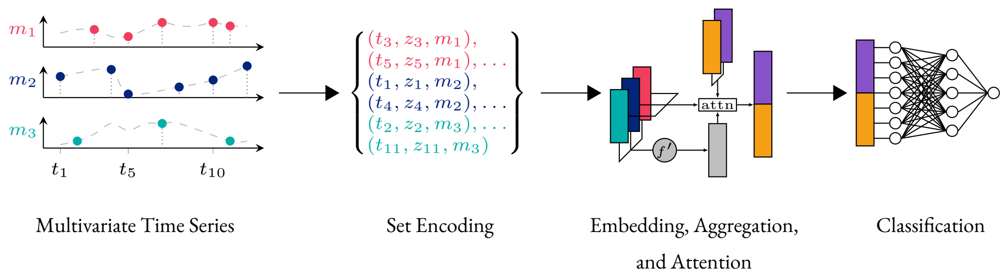

m_1
m_2
m_3
t_1
t_5
t_{10}
(t_3, z_3, m_1), $(t_5, z_5, m_1), \dots$ (t_1, z_1, m_2), $(t_4, z_4, m_2), \dots$ $(t_2, z_2, m_3), \dots$ (t_{11}, z_{11}, m_3)
attn
f'
Multivariate Time Series
Set Encoding
Embedding, Aggregation, and Attention
Classification

Figure 4.1: Overview of SeFT's architecture. The first panel shows a potential input, i.e. a multivariate time series with three modalities m_1, m_2 and m_3 which are sampled at irregular and unaligned times. We denote the j^{th} observation as a tuple (t_j, z_j, m_j), comprising of a time t_j, a value z_j and a modality indicator m_j. All observations are thus jointly summarized as a set of said tuples. Each set (and thus time series) is summarized using a small permutation-invariant set function f' which is concatenated to the embeddings of the individual set elements. A fixed size vector representation is computed by a weighted average using an attention mechanism with a static set of learnt query vectors leading to different representations which can focus on different aspects of the data. The representations are concatenated and passed to a classification MLP to predict the label of the time series.

mance, is highly scalable and improves over current approaches by almost an order of magnitude in terms of runtime. With SeFT, we propose to rephrase the problem of classifying time series as classifying a set of observations. We show how set functions can be used to create classifiers that are applicable to unaligned and irregularly-sampled time series, leading to favorable performance in classification tasks. Next to being highly parallelizable, thus permitting ready extensions to online monitoring setups with thousands of patients, our method also yields importance values for each observation and each modality. This makes it possible to *interpret* predictions, providing much-needed insights into the decision made by the model.

4.2.1 Proposed Method

This work focuses on the problem of time series classification of irregularly sampled and unaligned time series. We first define the required terms before describing our model.

Problem Statement and Notation

Definition 4.1 (Time series)**.** We describe a time series of an instance i as a set $\mathcal{S}_i$ of $M := |\mathcal{S}_i|$ observations s_j such that $\mathcal{S}_i := \{s_1, \dots, s_M\}$. We assume that each observation s_j is represented as a tuple (t_j, z_j, m_j), consisting of a time value $t_j \in \mathbb{R}^+$, an observed value $z_j \in \mathbb{R}$, and a modality indicator $m_j \in \{1, \dots, D\}$, where D represents the dimensionality of the time series.

We write $\Omega \subseteq \mathbb{R}^+ \times \mathbb{R} \times \mathbb{N}^+$ to denote the domain of observations. An entire D-dimensional time series can thus be represented as

$$\mathcal{S}_i := \{(t_1, z_1, m_1), \ldots, (t_M, z_M, m_M)\}, \tag{4.1}$$

where for notational convenience we omitted the index i from individual measurements.

We leave this definition very general on purpose, in particular allowing the length of each time series to *differ*, since our models are inherently capable of handling this. Likewise, we neither enforce nor expect all time series to be synchronized, i.e. being sampled at the same time, but rather we are fully agnostic to *non-synchronized* observations in the sense of not having to observe all modalities at each time point[1]. We collect all time series and their associated labels in a dataset $\mathcal{D}$.

Definition 4.2 (Dataset)**.** We consider a dataset $\mathcal{D}$ to consist of n time series. Elements of $\mathcal{D}$ are tuples, i.e. $\mathcal{D} := \{(\mathcal{S}_1, y_1), \ldots, (\mathcal{S}_N, y_N)\}$, where $\mathcal{S}_i$ denotes the i^{th} time series and $y_i \in \{1, \ldots, C\}$ its class label.

For an online monitoring scenario, we will slightly modify Definition 4.2 and only consider *subsets* of time series that have already been observed.

In the following, we will give a short illustrative example of how an irregularly sampled time series can be converted into its set representation. Let us assume that instance i represents measurements for an ICU patient, and consists of two channels the heart rate (HR) and the mean arterial blood pressure (MAP) which we refer to as modalities 1 and 2. Assume that $0.5\,\text{h}$ after admission of the patient, an HR of 60 beats per minute (bpm) was measured, and after $3\,\text{h}$ a value of 65 BPM. Further, let us assume the MAP was measured to be 80, 85, and 87 mmHg after $0.5\,\text{h}$, $1.7\,\text{h}$, and $2.5\,\text{h}$. According to the definition above, we can represent this time series as $\mathcal{S}_i := \{(0.5, 60, 1), (3, 65, 1), (0.5, 80, 2), (1.7, 85, 2), (3, 87, 2)\}$, where the observations are ordered to increase readability. It is important to note that in this representation there is *no* inherent ordering of the observations as they are elements of a set. Nevertheless, no information is lost, as we can construct a bijection between any set representation of a time series and the conventional time series representation by "scattering" the observation values z according to t and m. Our model, however, does not assume that all observations are stored or processed in the same ordering—this assumption was already shown [245] to have a strong impact on classification performance in some scenarios. Therefore, our model does not employ a "sequentialness prior": instead of processing a sequence conditional on previously-seen elements (such as in RNNs or

[1]We make no assumptions about the time values t_j and merely require them to be positive real-valued numbers because our time encoding procedure (see below) is symmetric with respect to zero. In practice, positive time values can always be achieved by applying a shift transformation.

other sequence-based models), it processes values of a sequence *all at once*—through encoding and aggregation steps—and retains all information about event occurrence times.

In our experiments, we will focus on time series in which certain modalities—channels—are not always observed, i.e. some measurements might be missing. We call such time series *non-synchronized*. Given the above definition, we consider a time series non-synchronized if there exists at least one time point t_j at which at least one modality is not observed, i.e. if the number of measurements associated with a time point is less that the number of modalities in the time series. Figure 4.1 gives a high-level overview of our method, including the individual steps required to perform classification.

Model description In the following, we describe an approach inspired by differentiable learning of functions that operate on sets [247, 267]. The following paragraphs provide a brief overview of this domain while describing the building blocks of our model. Specifically, we phrase the problem of classifying irregularly-sampled time series as learning a function f on a set of arbitrarily many time series observations following Definition 4.1, i.e. $\mathcal{S} = \{(t_1, z_1, m_1), \ldots, (t_M, z_M, m_M)\}$, such that $f\colon \mathcal{S} \to \mathbb{R}^C$, where $\mathcal{S}$ represents a generic time series of arbitrary cardinality and $\mathbb{R}^C$ corresponds to the logits of the C classes in the dataset. As we previously discussed, we interpret each time series as an unordered set of measurements, where all information is conserved because the observation time is included for each set element. Specifically, we define f to be a set function, i.e. a function that operates on a set and thus has to be *invariant* to the ordering of the elements in the set. Multiple architectures are applicable to constructing set functions such as Transformers [133, 239], or Deep Sets [267]. Given its exceptional scalability properties, we base this work on the framework of Zaheer et al. [267]. Intuitively, this amounts to computing multivariate dataset-specific summary statistics, which are optimized to maximize classification performance. Thus, we *sum-decompose* the set function f into the form

$$f(\mathcal{S}) = g\left(\frac{1}{|\mathcal{S}|}\sum_{s_j \in \mathcal{S}} h(s_j)\right) \tag{4.2}$$

where $h\colon \Omega \to \mathbb{R}^d$ and $g\colon \mathbb{R}^d \to \mathbb{R}^C$ are neural networks, $d \in \mathbb{N}^+$ determines the dimensionality of the latent representation, and s_j represents a single observation of the time series $\mathcal{S}$. We can view the averaged representations $1/|\mathcal{S}| \sum_{s_j \in \mathcal{S}} h(s_j)$ in general as a dataset-specific summary statistic learned to *best* distinguish the class labels. Equation 4.2 also implies the beneficial scalability properties of our approach: each embedding can be calculated independently of the others; hence, the constant computational cost of passing a single observation through the function h is scaled by the number of observations, resulting in a runtime of $\mathcal{O}(M)$ for a time series of length M.

Recently, Wagstaff et al. [247] derived requirements for a practical universal function representation of *sum-decomposable* set functions, i.e the requirements *necessary* for a *sum-decomposable* function to represent an arbitrary set-function given that h and g are arbitrarily expressive. In particular, they show that a universal function representation can only be guaranteed provided that $d \geq \max_i |\mathcal{S}_i|$ is satisfied. During hyperparameter search, we therefore independently sample the dimensionality of the aggregation space and allow it to be in the order of the number of observations that are to be expected in the dataset. Further, we explored the utilization of max, sum, and mean as alternative aggregation functions inspired by Garnelo et al. [83] and Zaheer et al. [267].

Time Encoding To represent the time point of an observation on a normalized scale, we employ a variant of *positional encodings* [239]. Preliminary results indicated that this encoding scheme reduces the sensitivity towards initialization and training hyperparameters of a model. This makes intuitive sense, as the ICU stays show a large variability in stay length such that using a single scalar for this long-tailed distribution could lead to issues during training. Further, the time encoding can allow the model to reason at different length scales more easily compared to using a single continuous variable encoding. Specifically, the time encoding we utilize converts the 1-dimensional time axis into a multi-dimensional input by passing the time t of each observation through multiple trigonometric functions of varying frequencies. Given a dimensionality $\tau \in \mathbb{N}^+$ of the time encoding, we refer to the encoded position as $x \in \mathbb{R}^\tau$, where

$$x_{2k}(t) := \sin\left(\frac{t}{\mathfrak{t}^{2k/\tau}}\right) \tag{4.3}$$

$$x_{2k+1}(t) := \cos\left(\frac{t}{\mathfrak{t}^{2k/\tau}}\right) \tag{4.4}$$

with $k \in \{0, \ldots, \tau/2\}$ and $\mathfrak{t}$ representing the maximum time scale that is expected in the data. Intuitively, this presents each time point through the perspective of multiple sine and cosine functions where each trigonometric function shows a different resolution with respect to the continuous-time variable. We select the wavelengths using a geometric progression from 2π to $\mathfrak{t} \cdot 2\pi$, and treat the number of steps and the maximum timescale $\mathfrak{t}$ as hyperparameters of the model. We used time encodings for all experiments, such that an observation is represented as $s_j = (x(t_j), z_j, m_j)$.

Attention-based Aggregation So far, our method permits encoding sets of arbitrary sizes into a fixed-size representation. For increasingly large set sizes, however, many irrelevant observations could influence the result of the set function. The *mean* aggregation function is particularly susceptible to this because the influence of an observation on the embedding shrinks proportionally to the size of the set. We thus suggest using a *weighted* mean in order to allow the model to decide which observations are relevant and which should be considered irrelevant. This is equivalent to computing attention over the set input elements, and subsequently, computing the sum over all elements in the set.

Our approach is based on *scaled dot-product attention* [239] with multiple heads $i \in \{1, \ldots, m\}$ in order to be able to cover different aspects of the aggregated set[2]. We define $a(\mathcal{S}, s_j)$, i.e. the attention weight function of an individual time series, to depend on the overall set of observations $\mathcal{S}$ and the value of the set element s_j. This is achieved by computing an embedding of the set elements using a smaller set function f', and projecting the concatenation of the set representation and the individual set elements into a d-dimensional space. Specifically, we have $K_{j,i} = [f'(\mathcal{S}), s_j]^T W_i$ where $W_i \in \mathbb{R}^{(\mathrm{im}(f')+|s_j|)\times d}$ and $K \in \mathbb{R}^{|\mathcal{S}|\times d}$. Furthermore, we define a matrix of query points $Q \in \mathbb{R}^{m\times d}$, which allow the model to summarize different aspects of the dataset via

$$e_{j,i} = \frac{K_{j,i} \cdot Q_i}{\sqrt{d}} \qquad \text{and} \qquad a_{j,i} = \frac{\exp(e_{j,i})}{\sum_j \exp(e_{j,i})}$$

where $a_{j,i}$ represents the amount of attention that head i gives to set element j. The head-specific row Q_i of the query matrix Q allows a head to focus on individual aspects (such as the distribution of one or multiple modalities) of a time series. For each head, we multiply the set element embeddings computed via the function h with the attentions derived for the individual instances, i.e. $r_i = \sum_j a_{j,i} h(s_j)$. The computed representation is concatenated and passed to the aggregation network g_θ as in a regular set function, i.e. $r* = [r_1 \ldots r_m]$. In our setup, we initialize Q with zeros, such that at the beginning of training, the attention mechanism is equivalent to computing the unweighted mean over the set elements.

Overall, this aggregation function is similar to Transformers [239] but differs from them in a few key aspects. Commonly, Transformer blocks use the information from *all set elements* to compute the embedding of an individual set element, leading to a runtime and space complexity of $\mathcal{O}(n^2)$. By contrast, our approach computes the embeddings of set elements independently, leading to lower runtime and memory complexity of $\mathcal{O}(n)$. This is particularly relevant as set elements in our case are *individual observations* so that we obtain set sizes that are often multiples

[2]Since we are dealing only with a single instance (i.e. time series) in this section, we use i and j to denote a *head* and an *observation*, respectively.

of the time series length. The scalability of transformer-based architectures is a field of ongoing research and since the composition of this work has progressed significantly [227], such that modern developments might actually make the application of attention architectures to sets of observations feasible. In our preliminary experiments, we observed that incorporating interactions into the embedding of individual set elements leads to convergence issues and instabilities during training rendering the *permutation equivariant* formulation of set functions by Zaheer et al. [267] inapplicable in our context.

Online Monitoring Scenario In the context of medical time series, it is often desired to construct a model which can be applied in an online scenario, i.e. where data is being passed into the model and it gives a new prediction at each time point. In order to ensure that a model can handle both the low data regime where only very few data points are available and the regime where multiple hours of data have been recorded, it is important to train the model on all potential data availability scenarios. While this could be achieved by simply cropping the whole time series into increasingly long segments and training the model on each of these sequences individually, this is highly inefficient as it prevents the reusing of the computations of shorter sequences when computing longer ones. This is especially critical in our case as part of the sequence embedding can be computed individually for each time point without being dependent on other computations.

To allow a prediction on a per-time-point basis which enables the application to online monitoring scenarios, we can rearrange the computations of the weighted mean in our algorithm. This allows us to compute these in an online fashion following

$$\begin{aligned} f(\mathcal{S}_i) &= \sum_{j \leq i} \frac{\exp(e_j)}{\sum_{k \leq i} \exp(e_k)} h(s_j) \\ &= \frac{\sum_{j \leq i} \exp(e_j) h(s_j)}{\sum_{k \leq i} \exp(e_k)}, \end{aligned}$$

where we omit the head indices for higher readability. In this case, both the numerator and denominator can be computed in a cumulative fashion which allows us to reuse the computation of $h(s_j)$ from previous time points. In practice, we additionally need to subset these predictions to select the last set element associated with each time point at which we would want to predict.

Loss function If not mentioned otherwise, we choose h and g in Equation 4.2 to be *multilayer perceptron* deep neural networks, parametrized by weights θ and ψ, respectively. We thus denote these neural networks by h_θ and g_ψ; their parameters are shared across all instances per dataset. Our training setup follows Zaheer et al. [267]; we apply the set function to the complete time series, i.e. to the set of all observations for each time series. Overall, we optimize a loss function that is defined as

$$\mathcal{L}(\theta, \psi) := \mathop{\mathbb{E}}_{(\mathcal{S},y)\in\mathcal{D}} \left[\ell \left(y; g_\psi \left(\sum_{s_j \in \mathcal{S}} a(\mathcal{S}, s_j) h_\theta(s_j) \right) \right) \right],$$

where $\ell(\cdot)$ represents a task-specific loss function. In all of our experiments, we utilize the binary cross-entropy loss in combination with a sigmoid activation function in the last layer of g_ψ for binary classification.

4.2.2 Experimental setup

In the following we will detail the datasets and experimental setup used to compare the individual models to each other.

Datasets and Tasks

In order to benchmark the proposed method, we selected three datasets with irregularly-sampled and non-synchronized measurements. We are focusing on two tasks (1) patient mortality prediction, which represents a whole time series classification task and was evaluated on two datasets and (2) sepsis[3] onset prediction which an online time series classification prediction task where the model must predict a label for each time point using the information that was available until then.

For the first task, we utilize the MIMIC-III dataset [113], which is a widely-used and freely-accessible dataset of distinct ICU stays. The median stay length is 2.1 d and a wide range of physiological measurements are obtained at a resolution of 1 h. Laboratory test results, collected at irregular time intervals, are also available in the dataset. Instead of applying our own preprocessing pipeline in order to extract and harmonize value from the large MIMIC-III database, we rely on a set of preprocessed data and tasks defined by Harutyunyan et al. [91]. Here the authors design a mortality endpoint that should be predicted using data from the first 48 h of the patient stay. In contrast, as we design methods specifically for irregularly sampled time series, we omit the binning step of their preprocessing pipeline and apply additional filtering[4]. Overall, the dataset contains 21000 stays with a prevalence of death of approx. 10 %. We refer to this dataset and task combination as M-Mortality.

[3] an organ dysfunction cause by a dysregulated host response to infection

[4] This was necessary as some patients had high-resolution recordings which are incompatible with the rest of the dataset and lead to problems when processing them with the methods presented here. We conjecture that these were missed in the design of the preprocessing pipeline due to the binning step.

Additionally, we rely on a further dataset for mortality prediction, the Physionet 2012 challenge dataset [86], which we refer to as P-Mortality, contains 12000 ICU stays each of which lasts at least 48 h. Here up to 37 time series variables are measured and some are only collected when required and thus might not be available for each stay.

Finally, for the online sepsis prediction task, we use the dataset provided by Reyna et al. [194] as part of the Physionet 2019 Sepsis Early Prediction Challenge. It contains observations from over 60000 intensive care unit patients from three different hospitals and up to 40 variables which are binned into hourly measurements. Each hour is further associated with a binary label that indicates whether sepsis occurred following a recently suggested clinical definition [215]. The labels are propagated into the past, such that a positive label corresponds to the onset of sepsis within the next 6 h and is kept positive after the sepsis onset. Reyna et al. [194] additionally provide a new score to capture the clinical utility of a predictor, which also incorporates if a prediction takes place early enough for a doctor to potentially intervene. This utility score can be normalized to give rise to U_{norm} where a perfect classifier (according to the utility measure) receives a score of 1 and a model which does not predict at all a score of 0. We refer to this task as P-Sepsis.

Comparison Partners, Training and Hyperparameter Selection

We compare our method to the following six approaches: 1. GRU-simple [47] 2. GRU-Decay [47] 3. Phased-LSTM [167] 4. Interpolation Prediction Networks [219] 5. Transformer [239] and 6. Latent-ODE [200]. All methods besides Latent-ODE were implemented in the same code base and underwent an unbiased hyperparameter optimization process. In the case of Latent-ODE, this was unfortunately not possible as the runtime for this approach was considerably higher than any of the other methods considered. We thus note that the performance of Latent-ODE should rather be interpreted as an overly pessimistic performance estimate.

To mitigate the problem of unbalanced datasets, all models were trained on balanced batches of the training data rather than utilizing class weights. This was done in order to not penalize models with a higher memory footprint[5]. Due to oversampling, the notion of an epoch is different from the common understanding. In our experiments, we set the number of optimizer steps per epoch to be the minimum of the number of steps required for seeing all samples from the majority class and the number of steps required to see each sample from the minority class three times. The training was stopped after 30 epochs without improvement of the area under the precision–recall curve (AUPRC) on the validation data for the mortality prediction tasks, whereas balanced

[5]These models would only allow training with small batch sizes, which combined with the unbalanced nature of the datasets would lead to high variance in the gradients.

accuracy was utilized for the online predictions scenario[6]. The hyperparameters with the best overall validation performance were selected for quantifying the performance on the test set. The train, validation, and test splits were the same for all models and all evaluations.

Hyperparameters were optimized by uniformly sampling 20 parameters according to a predefined hyperparameter grid. Evaluation on the test set was performed after restoring the model to the state with the best validation AUPRC / balanced accuracy. The final performance was derived by averaging the test performance of 3 independent runs for each model.

4.2.3 Results

Mortality Prediction Task

Table 4.1: Performance comparison of methods on mortality prediction datasets. "AUROC" denotes the area under the Receiver Operating Characteristic (ROC) curve; "AUPRC" denotes the area under the precision–recall curve. Evaluation metrics were scaled to 100 in order to increase readability. † denotes that the performance could be underestimated due to limited hyperparameter tuning compared to other methods.

Dataset	Model	Accuracy	AUPRC	AUROC	s/epoch
M3-Mortality	GRU-D	77.0(15)	**52.0(8)**	**85.7(2)**	133(8)
	GRU-Simple	78.1(13)	43.6(4)	82.8	140(7)
	IP-Nets	78.3(7)	*48.3(4)*	83.2(5)	81.2(85)
	Phased-LSTM	73.8(33)	37.1(5)	80.3(4)	166(7)
	Transformer	*77.4(56)*	42.6(10)	82.1(3)	*20.1(1)*
	Latent-ODE †	72.8(17)	39.5(5)	80.9(2)	4622
	SeFT-Attn	**79.0(22)**	46.3(5)	*83.9(4)*	**14.5(5)**
P-Mortality	GRU-D	80.0(29)	*53.7(9)*	**86.3(3)**	8.67(49)
	GRU-Simple	*82.2(2)*	42.2(6)	80.8(11)	30.0(25)
	IP-Nets	79.4(3)	51.0(6)	*86.0(2)*	25.3(18)
	Phased-LSTM	76.8(52)	38.7(15)	79.0(10)	44.6(23)
	Transformer	**83.7(35)**	**52.8(22)**	**86.3(8)**	**6.06(6)**
	Latent-ODE †	76.0(1)	50.7(17)	85.7(6)	3500
	SeFT-Attn	75.3(35)	52.4(11)	85.1(4)	*7.62(10)*

Table 4.1 shows the results of all models on the two mortality prediction tasks. We highlight the best-performing method for each metric in bold and the second-best in italics. SeFT exhibits competitive performance and in terms of AUPRC the approach ranks among the best methods for all tested conditions. For M3-Mortality, we outperform the Transformer both in terms of runtime and in terms of performance measured in AUPRC. Both GRU-D and IP-Nets show better performance in this case while also exhibiting considerably higher runtime. This favorable trade-off in terms of runtime and AUPRC is highlighted in Figure 4.2.

[6]This distinction was made as the utility score of the sepsis prediction task relies on a binary prediction and not on a continuous probability estimate. In order for hyperparameter selection and evaluation to be more in line we thus also selected a metric that quantifies the quality of a binary prediction.

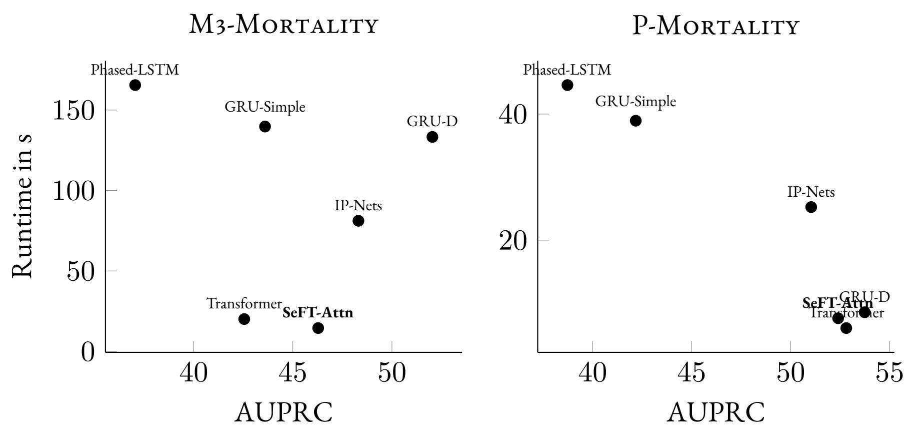


Figure 4.2: Runtime vs. AUPRC trade-offs for all methods on the two mortality prediction tasks. Latent-ODE is not shown as its runtime is significantly higher compared to the other models.

Online Sepsis Prediction Task

Table 4.2: Results of the online prediction scenario on the P-Sepsis task. The dataset is highly imbalanced, such that we only report measures that are sensitive to class imbalance. Further, if the results between the evaluation scenarios differ, we highlight results *without* masked future information in gray, and the performance achieved with masking with *. † indicates that the results might be underestimating the true performance due to limited hyperparameter tuning compared to the other methods.

Model	B-Accuracy	AUPRC	AUROC	U_{norm}	s/epoch
GRU-D	57.4(2)	5.33(39)	67.4(12)	12.6(11)	*72.3*
GRU-Simple	**71.0(14)**	**6.10(75)**	**78.1(15)**	**26.9(41)**	116
IP-Nets	87.1(9)	29.4(21)	94.1(4)	62.2(13)	253
IP-Nets *	63.8(9)	5.11(80)	74.2(12)	−11.9(40)	253
Phased-LSTM	67.5(17)	*5.54(91)*	75.4(13)	20.2(32)	192
Latent-ODE †	62.4(1)	11.4(21)	64.6(7)	12.3(10)	1872
Transformer	91.2(2)	53.4(56)	97.3(2)	71.3(14)	**28.5**
Transformer *	53.6(17)	3.63(95)	65.8(37)	−43.9(100)	**28.5**
SeFT-Attn	*70.9(8)*	4.84(22)	*76.8(9)*	*25.6(19)*	77.5

In order to verify if our approach is applicable to online monitoring scenarios in the clinic, we evaluated performance on the Physionet 2019 Online Sepsis prediction challenge. In this scenario, both the Transformer and IP-Nets yield the highest performance and outperform all other methods by a large margin. These results are even much better than those submitted to the challenge in the first place, which reached a maximum score of 0.36 on the test set [194]. As this seems highly suspicious we extended the evaluation to verify if the models relied on future information during test time by running the model on cropped versions of the time series, where for prediction at a time point only the information up to that point is provided. We included these additional results and mark them with *. To our surprise, it is clearly evident that the high performance of these models was due to the leakage of future information during the evaluation

For IP-Nets, information can leak through the non-parametric imputation step prior to the application of the downstream recurrent neural network. It is infeasible to train the vanilla IP-Nets approach on slices of the time series up until the time point of prediction, as we cannot reuse computations from previous imputation steps. While it would be possible to construct an IP-Nets variant that does *not* rely on future information during the imputation step, for example using smoothing techniques, we deem this beyond the scope of this work.

Similar effects occur in the case of the Transformer: While observations from the future are masked in the attention computation, preventing access to future values results in a detrimental reduction in performance. Even though the source of dependence on future values is quite probable to reside in the layer normalization applied in the Transformer model, the performance drop can have multiple explanations, i.e. 1. the absence of future time points leads to high variance estimates of mean and variance in the layer norm operation, resulting in bad performance in the initial time points of the time series, or 2. the model actively exploits future information through layer normalization. This could for example be possible by the model looking for indicative signals in future time points and when present returning very high norm outputs. The signature of these high norm outputs can then, through the layer norm operation, be observed in earlier time points. While one could construct variants where the Transformer model can by no means access future information, for example by replacing the layer norm layer with an alternative normalization scheme [11, 169], we reserve a more thorough investigation of this issue for future work.

By contrast, our model does not contain any means of leaking future information into the prediction of the current time point and thus exhibits the same performance in both evaluation scenarios, while remaining competitive with alternative approaches. Surprisingly, the model with the highest performance in this scenario is GRU-Simple, which could be explained by the very regular sampling character of the P-Sepsis dataset. Here the measurements were already binned into hours, such that they cannot be considered completely irregularly sampled. This explains the high performance of GRU-Simple, as compared to models which were specifically designed to cope with the irregular sampling problem.

Investigation of Attention Patterns

While recently the usage of attention weights for interpretability and explanation of models has been criticized [32, 110, 213], the consensus on how valuable attention maps are for these types of investigations is not clear and seems to depend on the exact definition of explainability/interpretability [257]. While given the current state of research we cannot conclusively say that access to attention scores helps with the interpretability of models and their explanation we do want to

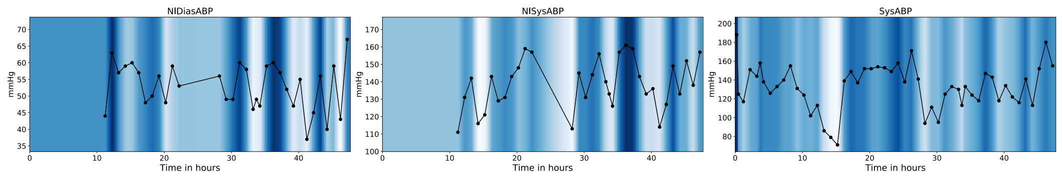


Figure 4.3: Visualizations of a single attention head on an instance of the P-MORTALITY dataset. We display a set of blood pressure variables that are most relevant for assessing patient stability: non-invasive diastolic arterial blood pressure (NIDiasABP), non-invasive systolic arterial blood pressure (NISysABP), and invasively measured systolic arterial blood pressure (SysABP). Darker colors represent higher attention values. In the invasive channel showing high time resolution (rightmost panel), our model attends foremost to the region after a sudden increase in blood pressure. In the non-invasive, intermittently observed channels, the model additionally focuses on regions of high observation density reflecting the clinician's concern.

highlight a *unique* property of our model which allows attribution of attention weights to individual observations. This is due to the existence of an information bottleneck in the attention mechanism and the elements of the attention mechanism being individual observations of measurements.

In the medical domain, this is of particular interest as a doctor would typically not only want to know the set of observations to be of relevance at a certain time for the prediction (as would be the case when we bin observations and apply attention mechanisms over the time points) but rather to know which *individual* observations are critical for the model. We show that our model supports this type of observation-based attribution using an example of clinically relevant variables from a patient time series which we overlaid with the attention values of our model in Figure 4.3.

We reviewed these records with our medical expert and find that for channels showing frequent and regularly-space observations, the model attends to regions that contain drastic changes in the values and to those measured when the patient is applied to the ICU. For instance, the invasive systolic blood pressure channel (SysABP), shows high attention for the first measurement at the beginning of the ICU stay which is above average and which dropped significantly after the first hour. In contrast, similarly high values do *not* trigger similar attention from the model indicating that the time point at which the measurement was obtained is of relevance. This could make sense as patients are often admitted to the ICU when they require to be stabilized after having an unstable state. Thus the time when the patient is admitted to the ICU can give the most information about the prior unstable state and might be the most informative value when a model tries to assess mortality. Interestingly, we further observe that our model additionally attends to regions of high observation density when manual intervention is required such as in the non-invasive blood pressure measurements (NISysABP and NIDiasABP). This could be a signal for the model that the doctor or nurse is paying a higher degree of attention to the patient which could also correlate with the patient's state. This of course could give rise to circularity issues as mentioned in the introduction of this chapter.

4.2.4 Conclusion and Discussion

In this section, I presented Set Functions for Time Series (SeFT), an approach that allows the classification of irregularly-sampled time series by interpreting time series as sets of observations. This avoids the necessity of binning or imputing time series prior to the application of downstream deep learning models. I show that the method yields competitive performance on real-world datasets and significantly improves runtime compared to other competitors as it is uniquely parallelizable. It is important to note though that it *does not* outperform state-of-the-art models, yet it shows that operating on the level of individual observations may be a promising avenue for the construction of models on irregularly-sampled data. In particular, making this class of models more performant could be achieved by understanding how attention-based models can be scaled to extremely long sequences [227] or how hierarchical aggregation structures could be implemented to allow multiple resolutions of processing in the model as often present in computer vision [140].

An alternative avenue for improving performance in this realm is the incorporation of additional priors into the model. While the shift in perspective leads to a model that is naturally applicable to irregularly-sampled time series, it does remove many of the priors (or in this context more often referred to as inductive biases) inherent to recurrent neural networks. Some inductive biases of sequential processing could be beneficial for downstream tasks. For example, one could imagine it to be easier for a recurrent model to learn how to compute the rate of change between two measurements compared to a model which does not assume any structure [262]. A further promising inductive bias for time series could be translational equivariance, which would lead to models that can only recognize patterns independent of the location in the time series. Recent research in time convolutional neural networks has indicated that this inductive bias could be beneficial in online monitoring [159] and time series classification scenarios [13]. While these approaches do not directly apply to irregularly-sampled time series and require preprocessing steps such as imputation and binning, some recent work shows how translational equivariance can be implemented in CNNs applied to irregularly-sampled domains [76]. It is generally possible to implement equivariance in the attention computation using relative position embeddings [216], yet this can also lead to scalability issues, especially when treating time series as sets of observations. In follow-up work, I thus further explored how translational equivariance can be implemented in the attention computation of scalable transformers [103]. Here I showed that it is possible to implement translational equivariance in scalable transformers by augmenting the feature space in which the attention is computed.

Finally, while we showed visualizations of attention patterns in order to highlight a unique property of our model of attending to individual observations, the interpretation of attention patterns has recently been criticized [32, 110, 213]. Nevertheless, others have criticized that attention can be useful for the explanation of predictions if tests are appropriately performed [257]. The interpretability of a model and its potential to explain why certain predictions were made can potentially determine a model's applicability to the clinic. When critical care decisions are made based on the model's predictions, it is important that clinicians have the possibility to understand why a prediction was made and potentially question it. This is linked to the development of general machine learning systems that are robust, trustworthy, and explainable and has emerged to become a subfield of its own [155]. Here many have called for the direct application of interpretable models instead of a post-hock explanation of black-box approaches [201], especially when a model is used to make high-stake decisions.

4.3 Investigating Hospital Transfer and Circularity

As mentioned at the beginning of this chapter, another important issue besides the form of medical time series data is the biases inherent to its generation. Treatment plans between hospitals and countries vary such that models which are trained on data from a specific origin will with a high probability not transfer to other locations. Further, the intervention of a doctor contains important information about what the doctor's expectations with respect to a patient are. A doctor would typically only give antibiotics if he suspects that the patient has a bacterial infection. Further, dependent on the *type* of antibiotics used the model might even be able to infer which type of infection the patient is suffering or which resistances the strain might have. This of course could lead to situations where the doctor and the model are waiting for an action from the other: The model is waiting for a signal from the doctor which would give a clear indication if the patient is in a critical state, whereas the clinician could expect the model to signal the onset of deterioration. Of course, this scenario would have a detrimental effect on a patient as potentially neither of the actors would intervene in the case of deterioration and further outside of the clinical setting evaluating models without these issues in mind would lead to overly optimistic performance estimates that cannot be met in the real world.

In the following, we will explore how to tackle these two issues when applying machine learning to the problem of sepsis early detection.

4.3.1 Sepsis Onset Detection

Sepsis is defined as an organ dysfunction caused by a dysregulated host response to infection which often threatens a patient's life [215, 222]. It remains a major threat to public health and is associated with high rates of mortality and morbidity and results in significant health costs [60, 105, 118, 182]. The infection and the associated dysregulated immune response lead to organ damage which increases for each hour in which sepsis is not recognized and appropriately treated ultimately leading to higher mortality of patients [74, 188, 214]. This makes it extremely important to treat sepsis in a timely manner in order to stop further harm from occurring and thus increase the patient's chance of survival [74]. Typically, sepsis is treated by resuscitation, organ support[7], and giving antibiotics in order to stop the progression of infection. One of the main issues in treating sepsis is that it is hard to diagnose as it is a complex and heterogeneous syndrome. In order to understand if a patient is undergoing an infection, doctors rely on the identification of bacterial species in the blood, which takes up to 48 h after the sample has been collected [179]. Thus in order to avoid severe organ damage doctors may often already induce treatment with antibiotics based on a suspicion of infection [74].

In order to provide alternative signals to doctors a series of biomarkers have been examined, yet until now with only limited success [29, 235]. While many promising development candidates exist [128, 271], there still is no clinical gold standard for accurate and early prediction of sepsis. As the search for reliable biomarkers continues, alternative data-driven approaches for deriving signals are becoming increasingly explored [78, 161]. These methods rely on the fact that patients in the intensive care unit are monitored continuously using bedside devices and are regularly checked upon by nurses to ensure their stability.

While the development of machine learning models for medical time series is still in its infancy and many models have not yet been applied in clinical case-control studies, retrospective studies are an important first step to assess their potential. They help to determine issues that should be resolved before transitioning into the clinic, some of which we have discussed at the beginning of this chapter. In the following, we will be concentrating on how we tried to avoid these issues and provide a detailed description of the study design where we evaluate the potential of machine learning methods for the early recognition of sepsis.

Problem Formulation In sepsis the earliness of a prediction is of elemental concern and thus the problem cannot be treated as a simple time series classification problem. Instead, we phrase sepsis detection as a time point-wise prediction task and require the model to score how probable it assumes sepsis is to occur in the next 6 hours after the current time point. We adapted this formulation from the Physionet 2019 sepsis early detection challenge [86], which provided labels at an hourly resolution using an endpoint definition similar to Sepsis-3. Formally, we define the instance x_i of the dataset to be composed of two matrices $\mathbf{v}_i \in \mathbb{R}^{l \times d}$ which denotes the

[7]Both terms refer to ensuring that the patient's physiological processes are maintained despite the critical state. One example is maintaining the patient breathing using manual ventilation.

observed values and $\mathbf{m}_i \in \{0,1\}^{l \times d}$ which represents if there are observations of this variable at this time point or if the value was imputed or filled. Further, each instance is associated with a label $y_i \in \{0,1\}^l$ which indicates that sepsis occurs within the next 6 h according to our endpoint definition. The positive label is kept up to 24 h after the onset of sepsis to incentivize the model to not miss a sepsis case completely, yet also to avoid spurious signals when the vital measurements of the patient might already have little to do with his prior sepsis infection due to having been treated on time.

Datasets and Preprocessing In order to assess how well-trained models can be transferred between hospitals in the case of sepsis early detection, we constructed a harmonized multi-center, multi-country cohort of septic and non-septic ICU patients from five publicly available datasets. The utilized datasets are (1) AUMC [230], (2) EICU [186], (3) MIMIC-III [113], (4) HIRID [161] and (5) Emory[8] . While the Emory dataset contains hourly resolved sepsis labels, these are not available for the other datasets and thus were implemented by us using the Sepsis-3 criteria [222]. We will detail the endpoint definition in the following.

In our preprocessing, we exclude patients under 14 yr and patients from hospitals where the final rate of sepsis is lower than 15 %, in order to avoid incorrectly labeled negative cases due to data missingness. This value represents an approximate lower bound on the number of sepsis cases reported according to previous studies [203]. Further, we exclude patient stays shorter than 6 h, with less than 4 distinct time points at which measurements where taken, with missing data for a window longer than 12 h, and patients where the onset of sepsis is outside of the ICU, earlier than 4 h into the ICU stay or after 168 h in the ICU. Due to the heterogeneity of the data, additional checks were implemented in order to ensure adequate harmonization. These include variable mapping, unit synchronization, and outlier filtering and were applied using the automated pipeline provided in the `ricu` package [19]. We inspected the distributions of biomarkers manually to ensure that they are similar across the five datasets.

To harmonize the datasets, we resample the data to an hourly resolution similar to the Physionet 2019 sepsis early detection challenge [194], where the value is determined by the median of all measurements that occurred during the hour. Further, to make models trained on one dataset applicable to other datasets we decide on a subset of variables that are plausibly relevant for sepsis, measured at regular intervals, and do not include direct information about the treatment of a patient, as this might lead to circularity issues. We thus excluded therapeutic variables such as antibiotics, intravenous fluids, or vasopressors from the data provided to the models to do predictions. This simultaneously addresses the circularity and conflicting treatment standards issues to some degree. For 4 out of the 5 datasets, we can define a large overlap of 59 variables which

[8]This dataset is a subset of the Physionet 2019 sepsis early detection challenge [194].

we refer to as the "core" variable set. In comparison, the Emory dataset only contains 35 of these variables and also contains predefined sepsis labels. We thus treat this dataset separately in our analysis.

Endpoint Definition In order to define the sepsis prediction endpoint, we follow the guideline of Sepsis-3 as closely as possible [222]. These are based on the co-occurrence of a suspected infection and an increase of the sequential organ failure assessment (SOFA) score [244] by two or more points. Hereby suspicion of infection was assumed if the patient was given antibiotics and fluid sampling was performed, whereby the earlier of the two is declared as the time point for a suspected infection. If the SOFA score increases by at least two points up to 48 h before or 24 h after suspicion of infection, the time at which the increase occurs is considered the *sepsis onset*. During the computation of the SOFA score, we set patients who were sedated to a Glasgow Coma Score (GCS) score of 15, which corresponds to no impairment of cognitive functions. Further, we do not include urine outputs that were registered 24 h before the admission to the ICU or within the first 12 h of being admitted to the ICU. This is due to the observation that in many cases, a large initial urine volume is measured directly at the ICU admission. The urine volume is involved in the computation of the renal component of the SOFA score, which accounts for any measurements of urine in the past 24 hours. Due to the initial high volume at ICU admission compared to all other measurements, this leads to a sudden drop in the total volume measured as soon as the ICU admission time point is not included in the computation anymore. This triggers a strong increase in the SOFA score which leads to very many presumably false positive SOFA increases requiring their exclusion for the determination of accurate sepsis onsets.

Due to differences between the datasets, the criterion of suspicion of infection is hard to implement and we needed to apply dataset-specific rules for its derivation. In the case of the EICU dataset, fluid sampling only occurs very infrequently and there are no reports on fluid sampling in the HIRID dataset at all. We instead use a heuristic in order to define the suspicion of infection criterion based on multiple antibiotics administrations. This approach was verified on the MIMIC-III and AUMC datasets for which both antibiotics and fluid sampling information are available. While the heuristic leads to significant overlap between the patients considered septic on the MIMIC-III dataset with a Jaccard similarity of 0.69, the definitions deviate to a strong degree on the AUMC dataset where a Jaccard similarity of 0.42 is achieved. We note that this is due to the fact that the AUMC dataset contains many surgical patients who are often given antibiotics in a prophylactic manner and find that if these are excluded the overlap is much higher (0.78) confirming the applicability of the devised heuristic.

Evaluation Strategy As the sepsis prediction task was formulated as an online prediction task, models are provided with new data and have to provide an estimate for the onset of sepsis given the additional information. We selected this setup in order to simulate a realistic deployment scenario during model development and took care that our models are also compatible with it (see Section 4.3.2 for further details). Additionally, we devise an evaluation strategy that can give a meaningful perspective of the model's performance in a clinical setting. For this, we evaluate the models using an encounter-focused strategy in terms of the Area under the Receiver Operator Characteristic (AUROC) curve and by computing median precision and earliness of the prediction at a fixed recall of 80 %. For both metrics, we use the unnormalized prediction scores and discard the first and last 0.5 percentiles to robustify the analysis to outliers.

The range of all scores is partitioned into 100 evenly spaced thresholds and for each threshold we consider the first score of each encounter that surpasses the threshold as the prediction for the whole time series. This part of the evaluation is set up, without considering the actual time at which the sepsis alarm is triggered, but solely to evaluate the performance in recognizing septic patients. We count positive predictions on septic patients as true positives and negative predictions where the scores never cross the currently selected threshold for control patients as true negatives. For the second part of the evaluation, we determine the threshold at which the model achieves 80 % recall in the detection of septic patients and report the precision obtained at this threshold as well as the median earliness, i.e. the number of hours before the sepsis onset at which the model first predicted a score higher than the previously determined threshold.

We constructed this evaluation scenario to be highly conservative in that it does not allow for repeated alarms which would increase recall but could also lead to alarm fatigue, i.e. the reduction of the meaningfulness of alarms due to overstimulation. While it is generally more challenging to reach high performance in this setting, it aligns our evaluation with clinical practice and provides a meaningful metric of performance for clinicians while ensuring that at most a single false alarm can be raised in a control stay. To ensure that performance metrics are comparable and the tasks derived for different datasets have a similar difficulty, we harmonize the prevalence of sepsis to the across-dataset average of 17 % by subsampling control patients in case the prevalence is higher and subsampling cases otherwise. We repeat this procedure 10 times and denote the final score as the average over these repetitions. In order to ensure we do not lose too many valuable positive cases in this procedure we verified that 98.3 % of the sepsis cases were included in at least one subsample.

4.3.2 Baselines and Machine Learning Models

Over the years a large number of clinical scores have been developed by medical practitioners in order to capture the patient's state with regard to certain important vital functions. These scores contain a lot of domain knowledge and thus represent viable baseline predictors in their own right.

In this study, we compare our models against the following clinical scores (1) national early warning score (NEWS) [115], (2) modified early warning score (MEWS) [224], (3) systemic inflammatory response syndrome (SIRS) [22], (4) sequential organ failure assessment (SOFA) [244] and (5) quick sequential organ failure assessment (qSOFA) [222]. The Sepsis-3 definition subsumes the SOFA score and thus we expect it to be a competitive baseline as it is a core component of the label definition. When using the clinical baseline scores, we purposefully *allow* access to therapeutic variables that include treatment information, as these scores are static and cannot "wait" for doctor responses in order to make a prediction. Nevertheless, it is important to keep in mind that this does lead to some circularity issues as label definition and score are highly related.

Further, we compare two types of machine learning models in our study, machine learning approaches based on deep learning which are specifically designed for sequence prediction tasks, and classical machine learning approaches which require feature engineering prior to their application and the conversion of the time series prediction task into many individual timepoint-wise prediction tasks. For the former, we consider a modified variant of the transformer architecture [239] (attn) which we adapted for online prediction tasks, and an RNN based on the gated recurrent unit (GRU) architecture [51] (gru). In the latter, case we utilize gradient boosted decision trees [119] and LASSO/L1 regularized logistic regression (lr).

Due to the different structures and capacities of the two model classes, we implement additional feature engineering steps for the classical models. We construct a total of 1269 features, based on multi-scale look-back statistics (mean, median, variance, minimum and maximum) over the last 4, 8 and 16 hours, measurement indicators and also measurement counts. Additionally, we include domain-knowledge-based scores as features that mimic those typically used in the clinic in order to assess the patient's state. The scores include (1) the shock index [4], (2) the oxygenation index [178], (3) the SOFA score [244], (4) the SIRS score [22, 135] and (5) the MEWS. In contrast to the previous baseline clinical scores, we changed the definitions of the scores used as features. We do not include components depending on laboratory values and vitals which are not part of our readily measured, non-therapeutic input variables in order to avoid circularity issues between the feature and label definitions.

Deep Learning Models for Online Prediction The deep learning models in this study are applied directly to the hourly measurements which are augmented with additional measurement indicators. We do not impute the missing values for the deep learning models, as we argue that this can directly be learned by the model using a single linear layer in combination with the measurement indicators[9]. The GRU architecture can be readily applied to this scenario, as it

[9]It is easy to show that we can construct a bijection of the weights between both imputed and non-imputed scenarios which would lead to the exact same output over all inputs. Thus all functional mapping can be equivalently learned by both formulations.

resembles a recurrent neural network and thus its hidden state is updated recursively after each newly available data point. The hidden state is then used to predict the output of the model at that time point.

In contrast, Transformer models have not yet to our knowledge been applied to online monitoring scenarios as they lead to a series of issues. One of the core elements in the transformer architecture is the `LayerNorm` operation which normalizes the representations of the individual sequence elements using statistics computed over all elements of the sequence [9]. While the Transformer architecture typically masks information from future tokens in the decoder layer, the normalization statistics are still computed over the whole sequence making. While this could lead to issues when applying the model in the real world, it might also provide the model with a way of leaking information from the future via computed statistics. Independently, even if these issues can be ruled out by computing statistics in a cumulative manner this leads to other issues. Statistics on a small number of values will be of very high variance and could lead to problems during training and prediction time. In order to avoid complications along these lines, we replace the `LayerNorm` operation in the transformer with a previously suggested alternative formulation `ReZero` [11]. With `ReZero`, the layer normalization and residual connection used in conventional Transformer architectures are replaced with a residual where the component added to the vector is scaled using a scalar α. Thus the residual becomes

$$x_{i+1} = x_i + \alpha_i F(x_i), \tag{4.5}$$

where $F(x)$ is the transformation of the layer and x_i is the input to the layer. In order to avoid, the norm of the vectors from increasing with increasing depth, the authors of `ReZero` suggest initializing α_i with zero, such that the model in its untrained state is simply an identity function. In our context, `ReZero` has the benefit, that it completely removes the dependence of the normalization and residual step on the other elements of the time series and thus the transformation can be applied for each time point without knowing anything about the other elements of the sequence.

Experimental Setup

Each dataset is split into a development set composed of 90 % of the data and a held-out test set which comprises the leftover 10 %. The development set is further split into 5 randomly samples training and validation, where the training split comprises 90 % of the total available data and the validation split 10 % respectfully. For each split, we stratify according to the per-dataset determined sepsis prevalence. In order to select hyperparameters of the machine learning models, we train the models for all candidate hyperparameter combinations on the first training split and estimate generalization performance using the corresponding first validation split. After deriving the appropriate hyperparameters, the fit all models on all training portions of the random splits in order to derive an estimate of the variance of the models. For each trained model we then quantify the performance on the test split in order to derive the final generalization estimate.

For the deep learning models, we define an additional online validation split which is tracked during training in order to prevent overfitting. This split is a 10 % subsplit of the training data and the performance of the model is evaluated after each pass over the dataset (often referred to as an *epoch*). If the loss on this online validation split does not improve during the course of 20 epochs of training or if the number of epochs reaches 100 the training is stopped. The model is then evaluated based on the state where it achieved the highest performance on this online validation split. Further, we train the deep learning models using a positive weight dependent on the class imbalance i.e. the contribution of sepsis cases is scaled such that a single sepsis case would have the same loss impact as a negative case despite there being fewer sepsis cases in the data.

The hyperparameters of all models were selected based on the binary cross entropy loss of the prediction and the binary sepsis label evaluated on a time point level.

4.3.3 Results

In our study we examine how clinical scores compare to classical machine learning models and deep learning models, how well the internal validation performance transfers to other datasets, and finally which features the trained models find especially relevant. We will address these questions in the following sections.

Internal Validation of Sepsis Prediction Performance

In order to assess how well the problem of sepsis early prediction can be solved in the absence of distribution shifts, we evaluate the performance of all models on the test split of the same dataset they were trained on. The results for all machine learning methods and clinical baselines are shown in Table 4.3.

All machine learning approaches show higher performance than the clinical baseline scores in terms of AUROC and in terms of precision at 80 % recall. While in some cases the clinical scores show higher earliness, these also show significantly lower precision compared to the machine learning models. Here the SIRS and MEWS scores are especially early in predicting the sepsis onset on the EICU, HIRID, and MIMIC-III datasets. This seems reasonable for the SIRS score as it is designed to recognize septic shock and its associated high systemic inflammatory response. Further, SIRS is a core part of the Sepsis-2 definition [135] and some studies have shown its higher efficacy in detecting infection in the emergency department [82]. The results are more surprising for the MEWS score, where only very few studies have indicated its potential specifically

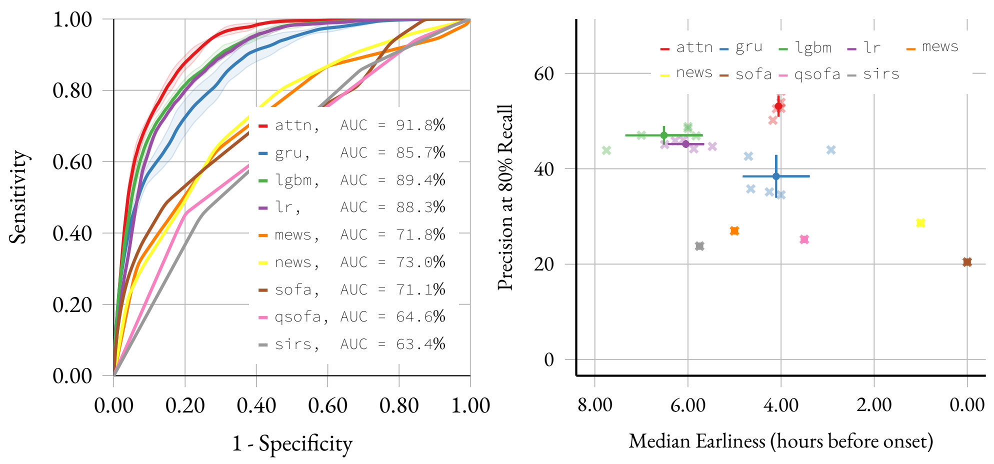


Figure 4.4: Predictive performance of models on AUMC evaluated as an internal validation, i.e. on the test split of the AUMC dataset. Left: Performance in terms of the Receiver Operator Characteristic. Right: Trade-off between earliness and precision at 80 % recall.

Table 4.3: Performance of machine learning approaches and clinical baselines when validated on the test split. Each sub-table contains the performance in terms of different metrics. Standard deviations were omitted for higher readability. The best performance for each dataset is highlighted in **bold**. "—" indicates that a value could not be computed due to the absence of necessary data.

Area under ROC curve

	attn	gru	lgbm	lr	mews	news	sofa	qsofa	sirs
AUMC	**91.8**	85.7	89.4	88.3	71.8	73.0	71.1	64.6	63.4
EICU	**80.3**	79.3	77.2	77.5	64.6	64.8	70.7	61.6	65.2
HIRID	83.4	82.5	**83.9**	81.1	56.8	65.0	76.1	54.2	60.9
MIMIC-III	**83.2**	81.8	83.1	80.0	60.9	65.3	69.3	56.6	60.9
Emory	**84.7**	82.5	78.7	84.3	—	—	—	—	—

Earliness at 80.0 % recall

	attn	gru	lgbm	lr	mews	news	sofa	qsofa	sirs
AUMC	4.06	4.11	**6.52**	6.05	5.00	1.00	0.0	3.50	5.75
EICU	4.63	4.65	6.03	4.19	**10.3**	9.25	3.75	8.00	8.00
HIRID	2.77	4.16	3.11	2.43	**6.80**	6.60	2.45	3.80	6.63
MIMIC-III	3.37	3.77	3.86	3.24	5.43	4.08	0.0	4.25	**6.88**
Emory	28.2	**31.5**	26.3	30.2	—	—	—	—	—

Precision at 80.0 % recall

	attn	gru	lgbm	lr	mews	news	sofa	qsofa	sirs
AUMC	**53.1**	38.4	47.0	45.2	27.0	28.6	20.4	25.2	23.8
EICU	**32.1**	30.8	29.5	29.3	20.0	21.7	24.4	23.2	24.7
HIRID	**36.4**	35.3	36.0	33.5	18.5	22.5	28.7	18.7	19.2
MIMIC-III	**35.7**	33.5	35.5	31.8	20.6	22.4	24.7	19.1	20.3
Emory	**39.4**	32.5	28.5	37.4	—	—	—	—	—

in the context of sepsis [104] and some even indicate that it might not be beneficial in this context at all [94]. Nevertheless, both scores show very low precision compared to all scores besides the SOFA score, where it is more strongly the case for the MEWS score. The clinical score with the overall highest average performance in terms of area under the ROC curve is the SOFA score which reaches an AUROC of 71.8 %, which is in line with its role in the endpoint definition. The fact that the earliness of the SOFA score is 0 for both AUMC and MIMIC-III and thus directly marks the onset of sepsis is further to be expected. It shows that at a threshold corresponding to 80 % recall a SOFA score alarm is uniquely triggered in sepsis patients at the time point at which it defined the sepsis onset. This is not a surprise, as the endpoint definition includes a large window of 96 h in which the first 2 point SOFA score increase is used to define the sepsis onset. Nevertheless, the low precision shows that also many non-septic patients exhibit large SOFA scores, which would lead to 80 % of the alarms being triggered incorrectly.

The attention model shows the highest detection performance in terms of AUROC on all datasets besides HIRID where the lgbm model outperforms. Interestingly, the GRU model often shows poor detection performance when compared to the classical models on the AUMC and MIMIC-III datasets. Among the machine learning models, the classical models lr and lgbm often show better or competitive early detection performance compared to the deep learning models (measured in terms of earliness), yet lower precision. The attention model outperforms all methods in terms of precision on all datasets and on the AUMC dataset even by a large margin of 6 %. Interestingly, the attn and gru models seem to show different trade-offs in terms of earliness and precision despite being trained using the same loss function. While the attn model is more precise in its predictions, the gru model typically recognizes sepsis earlier by approx. half an hour. A different trade-off can also be noticed between the classical and deep learning methods where the latter seems to put a stronger weight on precision over earliness.

In general, it is evident that there is no single model which clearly performs best in terms of all evaluation scenarios and datasets. While the attn model shows promising performance in detecting sepsis with high precision the classical models and some of the clinical scores show impressive early detection performance. The trade-off which of the two should be more beneficial in a clinical context is not clear and would have to be judged by clinical practitioners. It is important to note though, that earlier but less precise detections are not always desirable as they could induce alarm fatigue, where an individual alarm is given less attention due to its low precision.

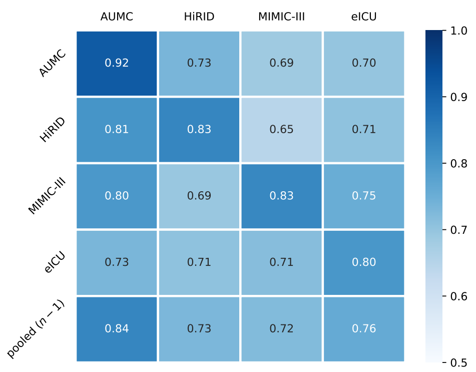
Finally, the results on the Emory dataset strongly differ from the results on other datasets, especially in terms of earliness. Here it is not totally clear how all models are able to achieve such high early detection performance while still showing relatively high precision. As we did not devise the endpoint definition it could be that the tasks show different difficulty levels and that this leads to high earliness on the dataset. Further, due to the reduced size of the variable set (only

35 variables instead of 59 variables are available for Emory), it could be that the models cannot uniquely differentiate onsets in a reliable manner and that they thus predict high scores also in the cases where the onset is much further away and where the model has not been incentivized to predict a positive label. Due to our evaluation scenario, this is then counted as an early onset while not contributing to a worse precision in the case of septic patients. Finally, it could also be that some cases can be recognized as septic already using the static variables. In this case, the models might predict sepsis onset already at admission.

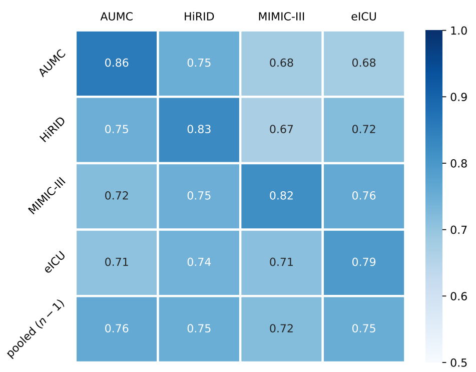
To further examine the performance of the models and baselines we concentrate on the results of the dataset with the highest average performance in terms of AUROC, AUMC. A more detailed perspective on the results for this case is thus shown in Figure 4.4. Here we see how different models occupy different regions of the earliness vs precision trade-off. In this particular example, it is also evident that the lr and lgbm models are very similar both in terms of detection performance and earliness vs. prediction trade-off. This partially makes sense as they were trained on the same set of hand-engineered features. In contrast, the attn and gru models show stronger differences, which we can attribute to the fact that both try to learn features from the raw data while having very different architectures. These differences in the architectures lead to differences in the types of features. While both attn and gru are universal function approximators [206, 266], the differences in architecture would lead to different functions being easier to learn due to the inductive biases encoded in the architecture. In additional experiments, we found that the higher performance on the AUMC dataset can be attributed to the surgical cohort which reaches 93.7 % compared to the medical cohort with 84.0 % in terms of AUROC. We could not reproduce these findings on the MIMIC-III dataset though, indicating that the surgical cohort of AUMC might contain distribution shifts in the feature or label distribution that make the task of onset detection easier.

External Validation of Sepsis Prediction Performance

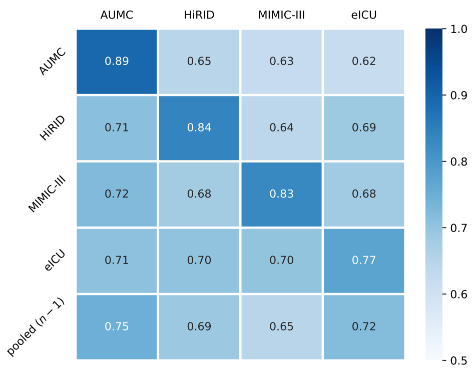
Due to the harmonization of the class imbalance across datasets, we can directly compare the performance of models trained on one dataset to their evaluation performance on other datasets despite the presence of prior shifts, i.e. different class prevalences across datasets. Further, as the models are also incentivized to predict according to the same prior class probabilities due to the positive weight in the loss term, training on different prevalences should not lead to any performance impacts when transferring between different datasets. A comparison of the detection performances in terms of AUROC is provided in Figure 4.5. It is clearly evident that all models perform worse when transferring to a dataset on which they were not trained. The degree to which the performance degrades is dependent on the model. Interestingly, the expectation that the very flexible models would perform worse on in the transfer task cannot be confirmed. There



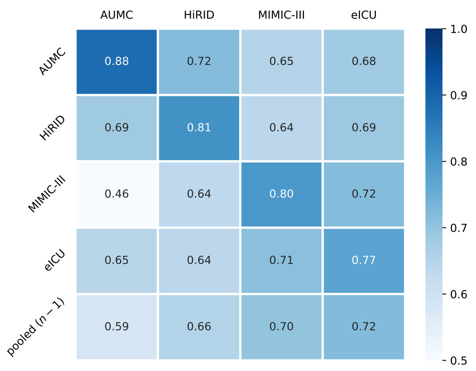
(a) attn



(b) gru



(c) lgbm



(d) lr

Figure 4.5: Pairwise comparison of transfer performance for machine learning models in terms of AUROC. The y-axis denotes the dataset on which the model was trained and the x-axis the dataset on which it was evaluated. Additionally, we incorporate a pooled model, which aggregated the scores from models trained on all datasets besides the one being evaluated. The pooling is implemented by taking the maximum of all predictions for a given time point.

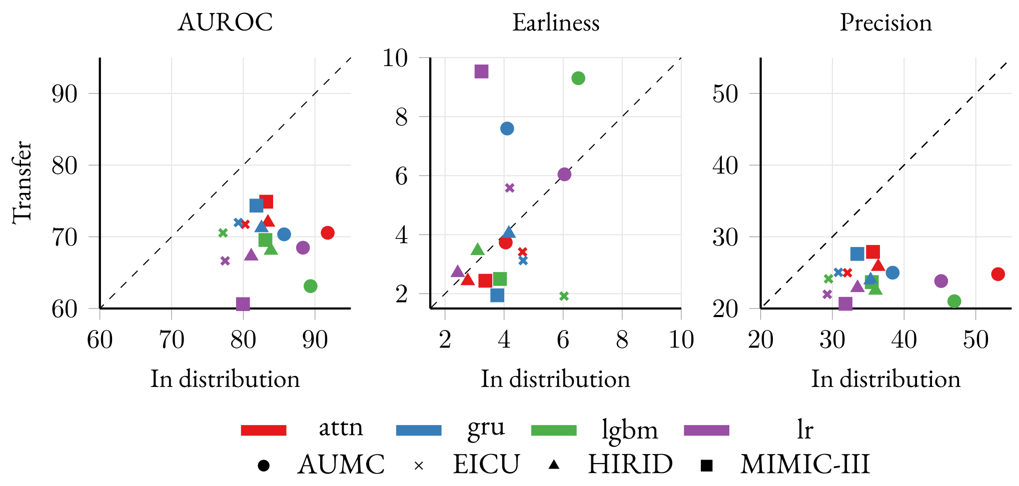


Figure 4.6: Average transfer performance of machine learning models compared to in-distribution performance in terms of AUROC, earliness, and precision. The x-axis shows the performance of the model on the training distribution, where the marked highlights which dataset was used for training. The y-axis shows the average performance on all other datasets (i.e. the datasets on which no training took place). The y-axis thus shows the average transfer performance to other datasets. EMORY was excluded from this analysis as its performance metrics (especially earliness) were found to be not comparable to the other datasets.

is not a single transfer scenario where the classical models show higher performance in terms of detection than the deep learning based models. Besides evaluating the individual models, we also construct an ensemble model, which pools the predictions of the same model trained on all other datasets. We can see that the ensemble approach performs favorably for all models and in the case of the deep learning models at least as well as applying a transferred model from any other dataset in most cases (the exception being transfer from MIMIC-III to EICU for the gru model). This is not the case for the conventional models, where for both lgbm and lr EICU to MIMIC-III transfer leads to better performance than the ensemble approach and there are many further cases where individual transfer combinations perform favorably. Nevertheless, while this highlights that ensembling predictions from multiple datasets can be beneficial, it only resembles a very first step and it should be further investigated which types of ensembling would be most beneficial. In a subsequent analysis, we further compare if detection performance is significantly different to the simple ensembling approach when training on all but one dataset and evaluating on the held-out data. The results are shown in Figure 4.8a and we did *not* observe a strong performance gap between the approaches.

In order to evaluate if any single model performs favorably in terms of generalization performance to other datasets, we compare the performance on the same datasets to the average performance obtained on all other datasets and show the results in Figure 4.6. In terms of AUROC the attn, gru and lgbm models show good transfer performance when being trained on the EICU, HIRID, and MIMIC-III datasets whereas in many cases the logistic regression model performs significantly worse for these datasets. When examining the trade-off between in-distribution vs. transfer performance, the AUMC datasets represent a striking outlier. Interestingly, the picture of the generalization performance of the models is flipped in this case, while attn, gru and lr models reach acceptable performance, the performance of the lgbm model is much lower. This could be related to our previous observation that the high performance on AUMC is linked to the surgical cohort and that there might be a distribution shift in the label or the features. While the lr model cannot pick up this distribution shift as it does not account for interactions between features and thus still achieves acceptable performance in the transfer setting, the lgbm model can account for more complex feature interactions and thus shows a significant drop in performance. The high performance of the deep learning models might be due to the preference for simpler functional mappings [237], yet finding the exact cause for this resilience to the distribution shift is beyond the scope of this study.

When examining the performance in terms of earliness and precision at 80 % recall some dataset model combinations lead to very similar earliness between the training dataset and the transfer scenario. These are always connected with a significant drop in precision, thus while the models detect sepsis onset for patients at comparable times they do raise more false positive alarms. Dependent on the transfer scenario the drop in precision is more or less pronounced for different models. Overall, the combination of the deep learning models and the AUMC dataset seems to lead to favorable transfer performance in the considered scenarios. It achieves both, an acceptable earliness of 3.5 h to 7 h and acceptable precision and detection performance. While training the deep learning models on the MIMIC-III dataset leads to higher precision of detections, the predictions are much later which can be detrimental in the clinical context. While the performance of AUMC is considerably lower than the performance on the training dataset when transferring to other datasets the deep learning models still reach highly competitive performance in the transfer setup.

Analysis of Feature Importance

In order to elucidate the role of individual variables and their predictability for sepsis we analyzed the attn model using Shapley values [147] which estimate the contribution of individual features to a model's prediction. We concentrate the analysis on the measurement values and discard count variables, missingness indicators, and derived features and provide the results of the analysis in Figure 4.7. According to the Shapley feature importance values, “Mean arterial pressure” and “Heart rate” contribute to the model's prediction most and suggest that the model has learned to incorporate values characterizing hemodynamic stability in its predictions. According

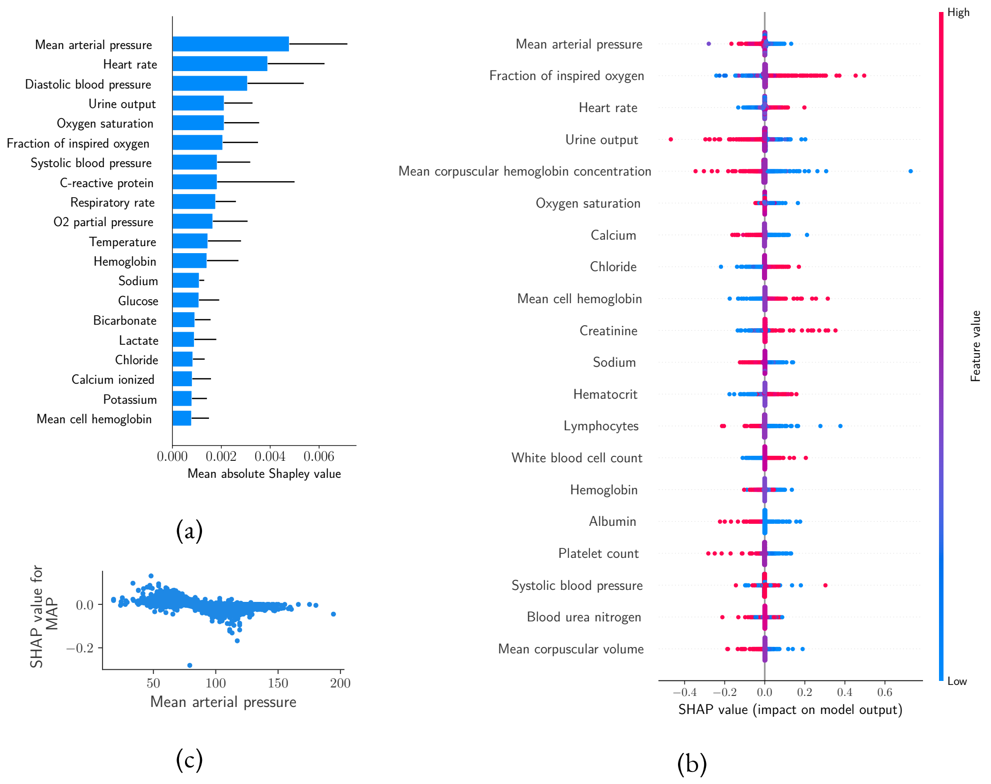
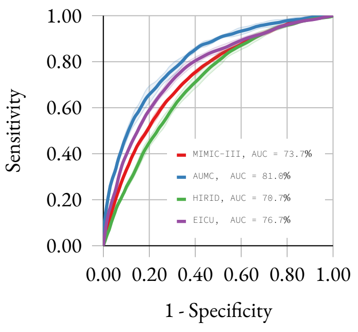
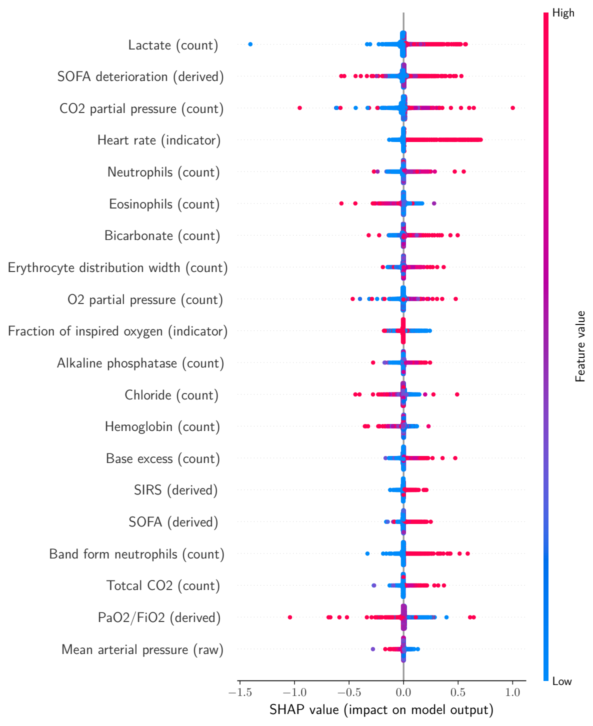
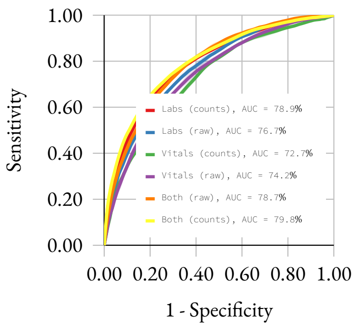


Figure 4.7: Exploration of feature importance via Shapley analysis of attn model. (a): Average absolute Shapley values over all datasets. The error bars represent the standard deviation across the datasets. The top 20 most important variables according to the Shapley score are shown where large values indicate a large contribution of the variable to the model's prediction and thus could be considered important for sepsis onset prediction. (b): Distribution of Shapley values for the EICU dataset. Positive values are associated with the model deeming sepsis onset in the next 6 h more probable, whereas negative values indicate the opposite. (c): Scatter plot of the most important variable "Mean arterial pressure" and its Shapley value on the EICU dataset.



(a)





(c)

(b)

Figure 4.8: Detailed explorations of model predictions. (a) The sepsis detection performance of the attn model when evaluated on a single dataset and trained on a pooled version of all other datasets. (b) The distribution of Shapeley values the attn model trained on EICU while also including count and derived features. (c) Ablation study of the attn model trained on feature subset of the MIMIC-III datasets.

to the attributions in Figure 4.7b, high heart rate, and low blood pressure are considered indicative of sepsis onset. This is in line with the definition of the SOFA score, where hypotension is a criterion for a higher score in the cardiovascular component and with the clinical definition of septic shock, which associates low mean arterial pressure with adverse outcomes [134]. Similar observations can be made for the variables “Platelet count” and “Creatinine” + “Urine output” which are included in the SOFA scores for the coagulation and renal components. Finally, it is important to note, that features do not always give high-magnitude Shapley values but are often only relevant in combination with other features. This is illustrated in Figure 4.7c where we can see that despite the trend of smaller Shapeley values for higher MAP the Shapley value remains at 0 for a majority of instances independent of the MAP value.

In further analysis, we noticed that count features were considered to be contributing strongly to the model predictions (see Figure 4.8b), which lead us to perform an ablation study to analyze their effect in more detail. For the ablation, we the attn model from scratch on the MIMIC-III dataset while only providing the raw measurements or the counts of either lab measurements or vitals and present the sepsis detection performance in terms of AUROC in Figure 4.8c. Interestingly, the model already achieves very high performance when trained only on the counts of lab measurements, whereas the performance suffers significantly when trained only on counts of vital signs. Further, training on the raw measurements of lab values leads to a more than 2 % decrease in performance compared to the counts alone. This indicates that despite removing all therapeutic variables and measurements with significant impact on the label definition much of the performance of the model is still determined by observing the clinician. Most of the lab measurements are non-automatic and are requested explicitly by doctors when they suspect a patient to be under elevated risk and would like additional information in order to decide on an appropriate strategy to combat a potential deterioration. This is in line with other publications which have found the measurement of lactate to be predictive of patient deterioration [108]. Generally, this shows how easy it is to design predictors which might not actually help in clinical practice due to their reliance on indirect doctor signals and further reinforces the issues of circularity mentioned at the beginning of this chapter.

4.3.4 Conclusion and Discussion

In this section I presented joint work on the first harmonized multi-center and multi-national ICU dataset for sepsis early prediction, which includes 5 databases from three different countries. We used this data to develop a large-scale set of sepsis label annotations and showed how these can be utilized to develop early warning systems based both on deep learning and hand-engineered features combined with classical machine learning approaches. In our large-scale evaluation, we highlight that a deep learning model based on the transformer architecture can achieve very high performance in terms of detecting sepsis on a patient level and in terms of precision when requiring a recall of 80 %. Nevertheless, the classical approaches with hand-crafted features achieve on average *earlier* detection of sepsis although arguably with lower precision, which could lead to problems like *alarm fatigue* when applied in a clinical context. Overall, our analysis showed that classical models and deep learning models occupy different regions in the earliness vs. precision trade-off space and that the decision of which approach is more suitable for the clinic boils down to a decision if earliness of onset detection or precision of sounded alarms is of higher importance.

This large-scale study further allowed us to examine how well models trained on one dataset transfer to other datasets. Our results highlight that a deterioration of performance is always to be expected when transferring in such a context and that the deep learning methods seem to show favorable transfer performance despite the expectation that these would over-fit to features of a particular dataset. In contrast to common intuition, we found that the most simple model, logistic regression, actually degrades in performance more severely that its more complicated counterparts. In order to boost the performance of the models in the transfer setting we suggest the utilization of ensembles over models trained on the held-out datasets and to aggregate the predictions at each time point to the maximum of all models. We find that this approach improves sepsis detection performance and allows the pooling of information from multiple sites without requiring costly retraining on a pooled version of the data.

When examining the models with regard to feature importance, we find that our models deem variables to be important which have not yet been associated with predicting sepsis [161]. In particular, the fraction of inspired oxygen could represent a potentially valuable predictive variable that might warrant further investigation. Nevertheless, these associations should be taken with caution. As our ablations showed, the model might despite all effort put into harmonization and reducing circularity still be able to indirectly infer the clinician's state of mind by paying attention to how many lab measurements are requested. This could also be the case for the fraction of inspired oxygen which could represent a proxy variable for the patient's intubation or mechanical ventilation state. Our study highlights how difficult it is to design approaches that would result in clear clinical impact where algorithms support clinicians in their decision instead of merely trying to distill clinicians' actions into predictive scores.

Initial retrospective studies that investigate sepsis prediction have shown promising potential to impact clinical care and improve the current state by providing early warning systems that help identify patients in urgent need of attention [80, 159]. Yet the recent literature shows a more nuanced picture: While models designed for the application in the clinic are being developed rapidly [36, 218] criticism of the applicability of these approaches in the clinical context and the lack of validation to other hospitals has been growing [161, 258]. The definition of the label often prevents the applicability of models to the clinic as many utilize billing codes that can represent an unreliable proxy [28, 77] and do not provide information about when sepsis actually occurred. Issues in the performance of widely applied models in external validation scenarios have further highlighted that transferability of models between hospitals is an issue [258]. This is due to the fact that external validation data is scarce and often requires significant additional manual work to ensure that the data are compatible and appropriately homogenized.

Our publicly available pooled ICU cohort promises to play an important role in assessing the external validation performance of newly developed models. If models are developed on our homogenized set of variables, these can be directly tested on 4 additional datasets besides the dataset on which a model was trained. Further, the combined dataset opens the door for more advanced machine learning approaches to ensure consistent performance across cohorts by for example leveraging recent developments in domain adaptation [73]. While most research at this intersection has focused on medical text [127] and medical imaging data [89], we conjecture that the transferability of models to multiple hospitals can also be improved by making the models more robust to individual treatment differences.

Finally, our study faces many limitations. While the overall size of the dataset is large, many patients and also sites were excluded in favor of a more homogeneous dataset with overall high data quality. These exclusions could have resulted in selection effects, which might be amplified by the fact that most of the patients in the cohort are Caucasian, such that the overall dataset might not well represent the variability that can be observed in an average hospital. Additionally, due to differences in data availability, we were required to construct two different definitions for suspected infection when implementing the Sepsis-3 criteria.

While our study strives to be as close as possible to an online monitoring scenario and to avoid any means of leaking therapeutic information in order to avoid circularity, we find that this can be an extremely difficult task. As the patient is already being continuously monitored by a doctor, it is almost impossible to out rule that the model does not simply try to reflect what the clinician might be suspecting. This leads to a large number of open issues that still need to be addressed. While intuitively, we would not want the model to have to operate on an imputed version of the medical times series data, it does seem the most straightforward approach in order to ensure that model does not account for additional sampling information and assumes a "missing at random" scenario.

Part III

Graphs

The final part of the thesis examines how to extend the learning of representations on graphs to incorporate a notions from multi-scale topology and how we can evaluate generated graphs.

5 Multi-Scale Topology for Machine Learning on Graphs

Graphs are a particular data structure and require specialized methods to be developed for machine learning. This is due to some of the inherent properties of graphs: they are unordered, relational structures, such that an individual node is best interpreted in the context of its neighborhood, i.e. the nodes it is connected to. They are of great importance for the analysis of networks such as social networks and relational databases of movies and for the pharmaceutical industry as small molecules can be represented as graphs [256]. Nevertheless, many models for predicting on graphs show limited expressivity and are unable to incorporate global information about the graph [34]. In the following chapter, I will introduce a new graph neural network for machine learning on graphs which takes inspiration from persistent homology. This allows our approach to harnessing multi-scale information present in the graph structure. In the first sections, I will give a brief introduction to graph-structured data, describe machine learning tasks on graphs and describe how graph neural networks are upper bound in expressivity by the Weisfeiler–Leman test. Afterward, I will introduce the proposed method.

This section is partially based on the following published work:
M. Horn*, E. D. Brouwer*, et al. "Topological Graph Neural Networks". In: *International Conference on Learning Representations*. 2022

5.1 Graph Structured Data

Graph structures are present in many real-world problems and allow the systematic inclusion of relational information into models. A graph is a set of objects (nodes/vertices) with some associated (potentially directional) relationship information between these objects (edges), a more formal definition can be found below.

Definition 5.1 (Graph)**.** A graph consists of a set of vertices V and a set of edges $E \subseteq V \times V$ in the case of directed graphs or $E \subseteq \{\{v_1, v_2\} : v_1 \in V, v_2 \in V\}$ in the case of undirected graphs. The combination of both is denoted as $G = (V, E)$.

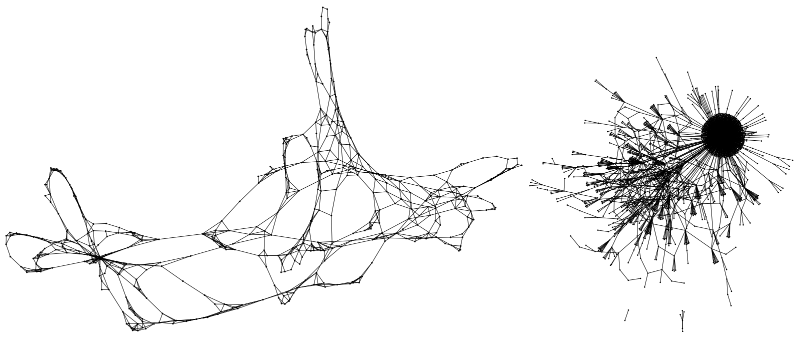


Figure 5.1: Visualization of two graphs from different datasets. The left represents an example from the DD dataset [26] and the right from the REDDIT dataset [263].

Graphs can be used to represent a variety of different structures that may or may not have physical resemblances in the real world. To give some examples, graphs can be constructed to represent proteins where the nodes represent the domains of the protein and the edges their physical proximity [26]. Further, two users on a social media platform can be represented as nodes and relationships among the users (for example *commented on*) as edges [263]. Visualizations of these examples can be found in Figure 5.1. As is evident from the visualizations, graphs can have very different structures, such that the existence of a "one size fits all" method remains questionable.

Often there is more information available about the nodes or edges of a graph. In this case, we call the graph an *attributed graph*. An attributed graph has the same properties as a graph, yet additionally defines attribution functions that map vertices and/or edges to attributes.

Definition 5.2 (Attributed Graph)**.** An attributed graph is a graph G equipped with at least one attribution function for either vertices $f_V : V \to \mathcal{V}_{\text{attr}}$ or for edges $f_E : E \to \mathcal{E}_{\text{attr}}$.

Most graph learning problems in machine learning deal with attributed graphs, where the presence of node attributes (often also referred to as node features) is predominant [164]. In practice, both $\mathcal{V}_{\text{attr}}$ and $\mathcal{E}_{\text{attr}}$ are required to be vector spaces, most prominently the space of d-dimensional real values is selected $\mathbb{R}^d$. If no node features are available it is common practice to use the degree of each node instead. When denoting algorithms on graphs, this text will refer to $n = |V|$ as the number of vertices in the graph and to $m = |E|$ as the number of edges in a graph.

5.1.1 Machine Learning Problems on Graphs

There are 4 main problem types or tasks that are tackled in the graph machine learning literature:

Graph classification/regression Prediction of a property of the overall graph. If for example, the graph represents the atoms of a small molecule, a typical task would be the prediction of molecular properties [256].

Node classification/regression Prediction of a property of a node. This task is often phrased as a semi-supervised learning task, where some node labels of a graph are given and the goal is to infer the missing ones.

Edge classification/regression Prediction of edge properties. This task has not been studied as extensively as the two above tasks. A common example of an edge classification problem is to characterize the nature of an interaction. For example, while people might be connected in a social graph the nature of this connection (friendship or rivalry) might not be evident. A further edge classification task can be the inference of missing edges in the graph [1].

Graph generation Generation of graphs that follow a defined distribution. Here a model is trained to mimic a distribution of graphs in order to generate new examples that are similar to the graphs provided. The most prominent application of this is the generation of small molecule graphs which should have certain properties.

In the following, we will be concentrating on graph and node classification, whereas later parts of this thesis examine the problem of graph generation in more detail.

5.2 Graph Neural Networks

Graph neural networks (GNNs) were first introduced in the work of Scarselli et al. [205] and have shown great success in graph applications. They have since been further extended methodologically leading to a whole zoo of GNN models [259, 273].

As the name indicates, a GNN is a neural network that operates on graphs, in contrast to the tensor inputs as common DNNs do. Further, instead of layers creating intermediate tensor representations, GNNs *output graphs as intermediate representations*. Typically, the connectivity of the graph (i.e. V and E) remains the same, yet the attributes associated with a node or edge may change.

Definition 5.3 (Graph Neural Network)**.** A graph neural network is an endomorphism g on the space of attributed Graphs. I.e. it maps between the space of attributed graphs, i.e. $g : \mathcal{G} \rightarrow \mathcal{G}$.

Usually, a GNN is comprised of multiple such transformations that are stacked and referred to as layers. The exact implementation of f varies greatly among different GNN architectures [259, 273]. Depending on the downstream task the attributed graph output of the GNN is transformed into a vectorial representation of the vertices (in the case of vertex / node classification and regression), of the edges (in the case of edge classification), or of the whole graph (in the case of graph classification). This final transformation is often referred to as a *readout function*.

5.2.1 Graph Convolutional Neural Networks

The simplest type of GNN is arguably the graph convolutional neural network, which will be used as an example of a GNN. In a graph convolutional neural network, the node features for each node are computed as a function of the node features of the neighboring nodes [122]. In particular, the node features of the $l+1$ layer are defined as

$$f_V^{(l+1)}(v) = \sigma\left(g\left(\sum_{v' \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{(d(v)+1)(d(v')+1)}} f_V^{(l)}(v')\right)\right) \tag{5.1}$$

where $\mathcal{N}(v)$ and $d(v)$ represent the neighborhood and degree of vertex v, g represents an linear map and σ a non-linearity, such as the sigmoid or ReLU function.

This relation is typically expressed using matrix notation, which leads to significant simplifications. Assuming the vertex attribution function of layer l maps to the space of d_l-dimensional real values $f_V^{(l)} : V \to \mathbb{R}^{d_l}$, define $H^{(0)} \in \mathbb{R}^{n \times d}$, $H^{(0)} = [f_V^{(0)}(v_1), f_V^{(0)}(v_2), ..., f_V^{(0)}(v_n)]^T$ to represent the matrix of node features / attributes and $A \in \{0,1\}^{n \times n}$, with $A_{i,j} = 1$ if $\{v_i, v_j\} \in E$ as the adjacency matrix of the graph. Then the above relation for the feature representation of the next layer can be written as

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right)$$

where $\tilde{A} = A + I$ represents the adjacency matrix with augmented self connections, as $\tilde{D}_{i,i} = \sum_j \tilde{A}_{i,j}$ and $W^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ is a layer specific trainable weight matrix.

Other Graph Neural Network Architectures There exist a plethora of GNN architectures developed for different use cases. Most of these architectures are not important for the general understanding of this thesis and thus will not be explicitly introduced. I kindly refer the reader to overviews by Battaglia et al. [15] and Wu et al. [259] for a more detailed introduction to the topic.

5.3 The Weisfeiler–Leman Test

In the following I will briefly introduce the Weisfeiler–Leman (WL) test [254] as it is necessary for understanding the limitations of graph neural networks. The WL test or algorithm is a test for graph isomorphism which is necessary but not sufficient in determining if two graphs are isomorphic. Thus, if two graphs are isomorphic the test will be positive, but a positive test outcome does not imply that the graphs are isomorphic. Testing graph isomorphism is a hard problem, rooted primarily in a graph's invariance to the reordering of nodes and the associated necessity for computing a canonical representation for graphs [107]. Interestingly, it has been shown that while the WL test is not sufficient, it does successfully test for isomorphism in a very broad class of graphs [10].

The algorithm operates based on an iterative node label refinement scheme (also referred to as "graph-recoloring", where node labels are considered the colors of nodes) which operates on undirected graphs [254]. The original version of the algorithm requires discrete node labels, although extensions of its kernel formulation to continuous node labels and weighted graphs have been proposed recently [232].

We will be concentrating on the 1-dimensional version of the WL-test [261], where we collect the neighborhood of a node and relabel the node using a hash function to provide it with a new label. First, every node in a graph is associated with a label or color. This is typically done using node attributes present in the data, yet it is also common to label each node with its degree if no node labels are available [24]. In each iteration of the algorithm, the labels of each node are collected and sorted to produce a consistent ordering. Finally, the label of the node itself is appended to the neighbor labels and the list is hashed using perfect hashing to produce a new label for the node as shown in Algorithm 2. The process is repeated for a predefined number of iterations or until the sorted label sequences of the two graphs diverge. If the label sequences never diverge, the graphs are possibly isomorphic, if the label sequences diverge the graphs are not isomorphic.

Algorithm 2 Iteration h of the 1-dimensional Weisfeiler–Leman graph recolouring scheme.

```
Require: $G = (V, E), f^{h-1} : V \to \mathcal{C}$
Returns: $f^h : V \to \mathcal{C}'$
    // Assign multiset label to each vertex based on neighbors
    $M_h(v) \leftarrow \{f^{h-1}(u) \mid u \in \mathcal{N}(v)\} \, \forall \, v \in V$
    // Sort multiset elements, prepend current label and store as string
    $s_h(v) \leftarrow (f^{h-1}(v), M_h(v)_1, \dots, M_h(v)_n)$
    // Map each string to a new compressed label using perfect hashing
    $f^h(v) \leftarrow \mathrm{HASH}(s_h(v)) \, \forall \, v \in V$
```

Higher-dimensional variants of this algorithm exist where instead of operating on the individual nodes, operations are performed on tuples of nodes and the neighborhood of both nodes is used to compute the new label of the tuple [165]. While higher-order WL tests can produce fewer false positive isomorphism predictions and retain the property necessity, they are computationally considerably more demanding. This is due to the necessity of enumerating all pairs, triplets, etc. on the graph, such that complexity grows exponentially with the number of nodes in the graph.

The properties of the Weisfeiler–Leman test have inspired the utilization of its underlying feature representation for the construction of Weisfeiler–Leman graph kernels [217]. While graph

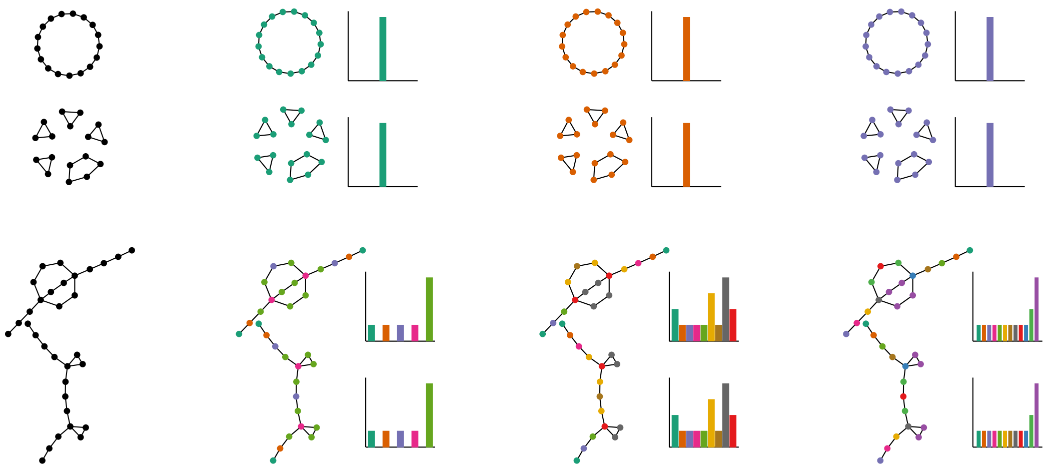


Figure 5.2: Examples of graphs that cannot be differentiated by the WL–test after 3 iterations when using node degrees as labels. Top row: Cycle graphs. Bottom row: Necklaces graphs. From left to right: 0, 1, 2, and 3 WL-iterations. Nodes are colored according to their new node labels derived by the WL relabeling scheme. The histograms correspond to the occurrence frequencies of the node labels. If label histograms coincide, the representative graphs cannot be differentiated using the WL test.

kernels have been an important step towards applying machine learning methods to graphs, they are not of importance for the understanding of this thesis and I thus refer the interested reader to Borgwardt et al. [24] for further information.

The Weisfeiler–Leman test provides a good basis for reasoning about the expressivity of graph algorithms. It has thus recently been used to characterize the shortcomings of graph neural networks and suggest paths to increase their expressivity [165, 261].

5.4 Limitations of Graph Neural Networks

Graph neural networks typically rely on iterative message-passing schemes and collapse information of each node's neighborhood into a new representation of the node. This is very similar to the Weisfeiler–Leman test [254], where a hash using the labels of the neighbors is computed to give rise to a new node label (for further information please refer to Section 5.3). A theoretical investigation of these similarities has shown that the expressivity of GNNs is upper bounded by the WL test [165, 261]. While it was initially hypothesized that Deep GNNs would be capable of detecting substructures in networks despite these shortcomings by leveraging high-order correlations in increasingly deep layers of the network, both theoretical and empirical evidence indicate that this is not the case [49].

A particular example of where the expressivity of the WL test falls short can be found in Figure 5.2. This is especially surprising as the graphs can be trivially differentiated by the human eye. Many such structures and graphs exist and have been characterized [7]. While this is not problematic in itself it can become crucial in cases where these properties are of high relevance for the prediction task [27]. This was highlighted in recent work by Dwivedi et al. [66], which showed that provably more powerful architectures do not necessarily lead to performance improvements on the datasets considered. It is unclear whether this is due to issues during the training of these more advanced models or if the invariances induced by GNNs are beneficial for many classification tasks.

5.4.1 Limitations of Graph Benchmarks

A further issue in researching GNNs is that it is not clear whether higher performance can be attributed to the exploitation of graph structure or simply to significantly more powerful models being applied to the vertex features. A recent study on graph kernels has shown, that in many cases a vertex histogram kernel can already achieve high predictive accuracy [24]. A vertex histogram kernel is the most simple way of including vertex features which raises the question of whether modern set function learning approaches (which do not include graph information) can compete with GNN architectures.

One alternative way of studying this issue is to ignore the vertex features altogether and thus evaluate how much information a model can extract from the graph structure alone. Recent benchmarking studies on GNNs have introduced node classification datasets based on stochastic block models (SBMs) where the node features are set to random values [66]. While this is a first step into elucidating individual performance contributions, more detailed investigations into these issues have not yet been conducted.

5.5 Extensions of Conventional Graph Neural Networks

As the limitations of GNNs have been known for a long time, there exist various extensions which aim at increasing the expressivity of neural networks that follow the message-passing scheme.

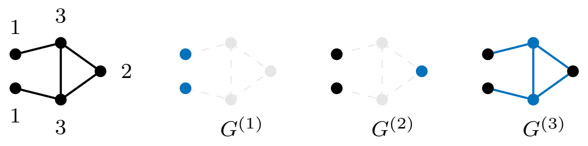
In their work Xu et al. [261] show that GNNs need to satisfy injectivity requirements to be able to reach 1-WL expressivity. Many approaches to go beyond the expressivity of the 1-WL–test build upon the fact that WL-iterations of order $k + 1$ are strictly more expressive than WL-iterations of order k for $k \geq 2$ [165]. Unfortunately, such k-order GNNs are computationally very expensive as they require the enumeration of all possible subsets of nodes. This has further been addressed by Maron et al. [151] which propose an alternative architecture that achieves 3-WL expressivity without requiring the enumeration of subsets, leading to improved runtime and space complexity. Further, Bouritsas et al. [27] propose an alternative approach to increase expressivity by including *graphlet features* that cannot easily be detected by GNNs [49]. Zhao et al. [272] follow a similar approach, yet harness local topological information using methods from the emerging field of topological data analysis (TDA) and use these features to reweigh the message-passing iterations of a graph neural network. Alternatively, Hofer et al. [97] suggest integrating the topological information of the graph at the level of the readout function, while showing that the filtration used to extract topological information from the graph can also be learned while training the GNN.

5.6 Persistent Homology on Graphs

While simplicial homology typically operates on simplices, these represent generalizations of graphs such that graphs can be interpreted as simplices and used in homology computations. More precisely, we can view a graph $G = (V, E)$ as a 1-dimensional simplicial complex [97], with the vertices V representing the 0-simplices and the edges E corresponding to the 1-simplices of the simplicial complex. In this context, the first boundary operator maps an edge to the formal sum of its two connected vertices and a cycle to the formal sum of its involved edges. In contrast to 0-dimensional topological features i.e. connected components, 1-dimensional topological features i.e. cycles cannot be destroyed when working purely on graphs as there are no higher dimensional features that could destroy them. Nevertheless, while it is possible to compute persistent homology of higher order by performing computations on simplicial complexes instead of regular graphs, this is typically not considered due to the significantly higher computational complexity associated with these computations.

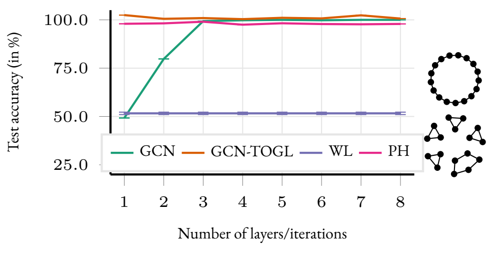
Graph Filtrations A common approach to the construction of a filtration on a graph $G = (V, E)$ is to select a function f that associates each vertex of the graph with a value $f\colon V \to \mathbb{R}$ and to ensure that the filtration is compatible with the graph my defining edge filtration values as $f(v_1, v_2) = \max\{f(v_1), f(v_2)\}$. Thus an edge can only be created when both of its vertices have already been created before or are created at the same time. Common vertex filtration functions are the degree filtration [99] or the vertices' contribution to the heat kernel [42].

As an example let us consider the degree filtration on a simple graph, where we visualize the new nodes and edges added to the graph highlighted, whereas black elements belong to structures that already exist prior to the step.

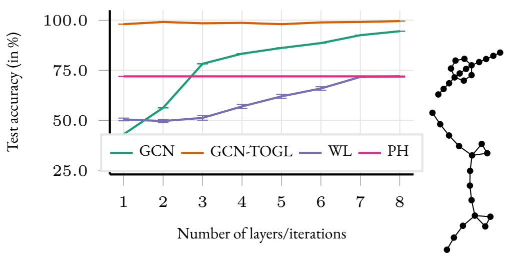


In this particular case, all edges are inserted at a filtration value of $f(v) = 3$ and we obtain the following 0-dimensional persistence diagram $\mathcal{D}_0 = \{(1,\infty),(1,3),(2,3),(3,3),(3,3)\}$. The essential persistence tuple $(1,\infty)$ indicates that this example contains only a single connected component and thus $\beta_0 = 1$. The one-dimensional persistence diagram only contains a single tuple $\mathcal{D}_1 = \{(3,\infty)\}$, and thus indicates that the graph contains a single cycle which is created at the filtration value of $f(v) = 3$ and that the Betti number for the graph is $\beta_1 = 1$. In this particular case, all edges are created at once and most are directly destroyed again due to the creation of the cycle.

5.7 Topological Graph Layer – TOGL



(a) Cycles



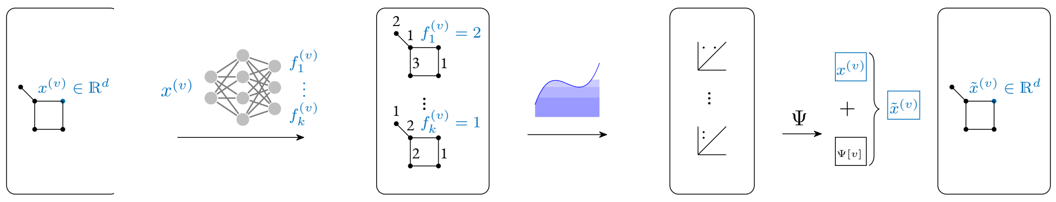
(b) Necklaces

Figure 5.3: We show the performance of (i) a GCN with k layers, (ii) our layer TOGL (integrated into a GCN with $k-1$ layers), (iii) the Weisfeiler–Leman (WL) graph kernel, and (iv) a method based on static topological features (PH). while TOGL can detect the difference in the graphs with only a single layer, other are only able to capture these structures at increasing depth.

In the following section, I will present and discuss Topological Graph Layer (TOGL), a drop-in replacement layer for GNNs. TOGL is motivated by the expressivity issues present in conventional GNNs of the Message Passing Neural Network type (see Section 5.4 for further elaboration). In contrast to many previous approaches, TOGL does not try to improve expressivity by climbing up the WL–test hierarchy but instead allows leveraging multi-scale topological information in the graph using persistent homology. We can show that our construction of TOGL retrains the expressivity of MPNNs while being able to differentiate additional graphs which can otherwise not be differentiated due to the upper bound of WL–expressivity.

Figure 5.3 provides us with a motivational example. Graphs that can be *trivially* classified by humans seem to represent extremely challenging examples for both the WL-kernel and common

GNNs. In contrast, the method presented in the following section is able to classify these graphs without having to rely on increasing depth. This is especially relevant as GNNs with large depth have often shown strong degradation in performance due to over-smoothing effects [48, 176]. An overview of the approach is depicted in Figure 5.4. TOGL is based on recent developments



(a) node features $x^{(v)}$ (b) node map Φ (c) k views of graph (d) filtrations (e) Persistence diagrams (f) aggregation (g) output $\tilde{x}^{(v)}$

Figure 5.4: Overview of the Topological Graph Layer (TOGL) method. (a) the node features $x_v \in \mathbb{R}^d$ and the graph structure are used as input, (b) a neural network Φ maps the node features to k filtration values, (c) each filtration can be interpreted as a different view on the graph, (d) the filtrations are computed based on the filtration values, (e) and give rise to k persistence diagrams. (f) The persistence diagrams are mapped into node level features using an embedding function Ψ and are combined with the input features in a residual manner giving rise to (g) the final output of the layer $\tilde{x}_v$.

for learning differentiable filtrations and propagating gradients through the persistent homology computation. The goal of the method is to allow the layer to (1) harness multi-scale topological information from computed persistence diagrams while (2) treating the filtrations as learnable mappings instead of statically engineered features and (3) incorporate this information into a hierarchy of arbitrary GNN layers.

We tackle points (1) and (2) by relying on recent developments in learning filtrations and multi-dimensional filtrations [39, 41, 97]. In order to address (3), it is necessary to ensure that the input and output of our method are compatible with existing architectures. In particular, we ensure compatibility with most GNNs by constructing a layer that takes an attributed Graph G with vertex features as input and returns an attributed Graph G where the new vertex features contain additional topological information. In the following, we clarify some notation to ensure the description of the method is unambiguous.

Notation We assume computations on an augmented graph $G = (V, E, f_V)$, where we refer to the vertices as $V = \{v_1, \dots, v_n\}$ and the vertex features as $X = [f_V(v_1), \dots, f_V(v_n)]^\top$. The vertex features of an individual node v are referred to as $x_v := f_V(v)$. Further, we denote the calculation of persistence diagrams of a graph G under some filtration f by $\mathbf{ph}(G, f)$. This will result in two persistence diagrams $\mathcal{D}_0, \mathcal{D}_1$, containing information about topological features in dimension 0 (connected components) and dimension 1 (cycles). While the computation of higher-order topological features is possible, this is computationally much more demanding and will not be covered in this text.

Method description We define k vertex filtration functions on GNN-computed vertex features and denote the i^{th} filtration as $f_i : \mathbb{R}^d \to \mathbb{R}$ for $i = 1 \dots k$, where d is the output dimensionality of the previous GNN layer. Similar to [97], we consider the GNN prior to TOGL as part of the filtration function. As the number of nodes in the graph is finite, the image of f_i is also finite and results in a set of node filtration values which we denote as $a_i^{(0)} < \cdots < a_i^{(n)}$ where the superscript corresponds to the sorted position of each filtration value. The filtration values are used to construct a filtration on the graph $\emptyset \subseteq G_i^{(0)} \subseteq \cdots \subseteq G_i^{(n)}$, where $G_i^{(j)} = \left(V_i^{(j)}, E_i^{(j)}\right)$ with $V_i^{(j)} := \left\{v \in V \mid f_i(x_v) \leq a_i^{(j)}\right\}$, and $E_i^{(j)} := \left\{v, w \in E \mid \max\{f_i(x_v), f_i(x_w)\} \leq a_i^{(j)}\right\}$. Using these filtrations we compute sets of persistence diagrams $\text{pers}(G, f_i) = \{\mathcal{D}_{l,i}, \mathcal{D}_{l,i}\}$, where $\mathcal{D}_{l,i}$ corresponds to the l-dimensional persistence diagram of filtration f_i. While it is possible to compute higher dimensional topological features, in our work we concentrate 0 and 1-dimensional topological features. In order to fulfill requirement 3 it is necessary to map the persistence diagrams back to node representations, such that the topological information can be utilized by conventional GNN layers. We implement this by defining *embedding functions* for the set of persistence diagrams associated with each dimension $\Psi_l : \{\mathcal{D}_{l,1}, \dots \mathcal{D}_{l,k}\} \to \mathbb{R}^{n' \times d}$, where n' is the number of vertices for $l = 0$ and the number of edges for $l = 1$. For a more detailed description on how we map persistence diagrams back to vertices and edges, please refer to the following section. The whole process is visualized in Figure 5.4.

Mapping Persistence Diagrams to Nodes The cardinality of $\mathcal{D}_0$ is equal to the number of nodes n in the graphs and each tuple in the 0-dimensional diagram is naturally associated with the *vertex* that created it. We thus exploit this naturally arising bijective mapping in order to map the embedded representation of persistence tuples in $\mathcal{D}_0$ back to the paired vertices, similar to previous work in topological representation learning [163]. In contrast, the cardinality of $\mathcal{D}_1$ is the number of cycles. As we only compute persistent homology until dimension 1, cycles never get destroyed. Following previous work by Hofer et al. [99], we thus pair them with the maximum value of the filtration. Conventionally, we would consider a cycle associated with the edge that lead to its creation yet, as we are *not* incorporating models which run message passing on edges in our study, we cannot directly include the cycle information of this form into the hierarchy of GNN layers. Further, the mapping of cycles to their creation edges has generally been deemed unstable [17]. Thus in order to inject information about cycles into the computation, we need to aggregate the cycle representations to a representation of fixed dimension, which can then either be included in the representation of *all* nodes or be used as an additional output of the GNN besides the readout function prior to classification. This is necessary as the number of cycles can vary between graphs making it difficult to incorporate without prior aggregation. We follow the first strategy in the case of node classification and the second strategy for graph classification tasks.

Filtration computation and embedding functions We compute the k vertex filtrations using a single MLP $\Phi : \mathbb{R}^d \to \mathbb{R}^k$. This allows to share information across filtrations, reduce the complexity of the computation by requiring a lower number of operations and speed up computations due to increased parallelism. The filtration f_i is then defined by the projection of Φ to the i^{th} dimension, i.e. $f_i := \pi_i \circ \Phi$.

In order to embed the persistence diagrams generated by the filtration into a representation which can be used in downstream layers, we follow two strategies: 1. utilization of DeepSets [267], which allows the computation of differentiable set2set mappings, and 2. summary functions from the persistent homology literature, in particular the rational hat function [100], the triangle point transform, the Gaussian point transform and the line point transform [42] . The latter group of approaches creates a purely local representation of each tuple, often representing the distance of the persistence tuple to a point or line in some transformed space. In this case, the embedding of each tuple can be considered independent of the embedding of the other tuples and independent of the other filtrations thus $\Psi_l(\mathcal{D}_{l,0}, \ldots, \mathcal{D}_{l,k})[v] = (\Psi_{l,0}(\mathcal{D}_{l,0}[v]), \ldots, \Psi_{l,k}(\mathcal{D}_{l,k}[v]))$, where $\mathcal{D}_{l,k}[v]$ refers to the tuple of the persistence diagram of dimension l and filtration k which is mapped to vertex v. In contrast DeepSets, embed each tuple in the persistence diagram conditional on the other tuples and filtrations, giving them higher expressive power. Nevertheless as indicated by Hofer, Kwitt, and Niethammer [100], there is no guarantee that the learned functions are continuous and stable with regard to typical metrics on persistence diagrams[1]. Further, they argue that the universal approximation properties of DeepSets require fixed size sets if the set elements to come from an *uncountable* domain as is the case for tuples in persistence diagrams which are defined on $\mathbb{R}^2$. Nevertheless, in our context, we can assume that the number of vertices and edges of graphs on which our method is applied is upper bounded.

Finally, in order to avoid the creation of noisy bottlenecks due to the discrete persistent homology computation, we incorporate the information in a residual fashion, such that $\tilde{x}_v = x_v + \Psi[v]$, where we denote $\Psi[v]$ as the slice in the embedding function output corresponding to vertex v while accounting for the vertex matching strategy outlined above. This is further compatible with the experimental setup of Dwivedi et al. [66] where all graph neural network where extended to utilize residual connections in the computation in order to reduce over-smoothing effects.

[1]For example it is commonly agreed upon that points close to the diagonal of the persistence diagram should be given less weight as they are typically topological noise. This of course cannot easily be implemented in a learned set function setup.

5.8 Theoretical Considerations

In this section we will consider the theoretical properties of TOGL. In particular, we will examine how it is possible to compute gradients through the persistent homology computation and how TOGL compares to other GNNs in terms of expressive power. Finally, we will discuss the runtime properties of TOGL.

5.8.1 Differentiability

The computation of persistent homology is inherently discrete due to the discrete creation and destruction of topological features and the fact that this discrete matching might change when perturbing the inputs. Nevertheless, it is possible to show that the computation of gradient is at least locally possible. The argument is that there always exists an infinitesimal change to the filtration values ϵ that would not lead to changes. This can be ensured if we assume the filtration values to be unique and indirectly follows from the definition of filtrations where the codomain is the real numbers $f\colon \mathcal{V} \to \mathbb{R}$. In the real domain, we can always perturb a sequence of unique values $u_1, \dots, u_n$, such that their ordering after sorting $u_1 < \dots < u_n$ remains the same[2]. If the ordering remains the same, this means the filtration on the induced structure will remain the same and thus the computation of persistent homology should lead to the same selection of features. On a filtration level, this requires injectivity of the filtration function and uniqueness of the vertex features. We will formalize the intuition about this below.

Previous work has shown that the map $\mathbf{ph}(\cdot)$ is (locally) differentiable [81, 97, 163, 187] when the filtration function is injective. We will include the formal proof based on Hofer et al. [97] for completeness.

Theorem 5.1 (Differentiability of **ph** compuation)**.** Let f be a differentiable vertex filtration function $f\colon V \to \mathbb{R}$ with continuous parameters θ, and let Ψ be a differentiable embedding function of unspecified dimensions. If the function f is injective for a specific set of parameters θ' i.e. $f(v) \neq f(w)$ for $v \neq w$ and the set of vertices is composed of distinct elements then the map

$$\theta \mapsto \Psi(\mathbf{ph}(G, f)) \tag{5.2}$$

is differentiable at θ'.

Proof. For notational convenience let $y_i = f(\theta', v_i)$ represent the filtration value for vertex v_i. Further, let π denote the sorting permutation of the filtration values $(y_i(\theta'))_{i=1}^n$ such that $y_{\pi(1)}(\theta') < y_{\pi(2)}(\theta') < \dots < y_{\pi(n)}(\theta')$. By the assumption of unique elements in V and

[2]This is because the real numbers with the standard euclidean metric are a (separable) Hausdorff space.

injectivity of the filtration, the filtration values also have to be distinct. Thus, there exists a neighbourhood around θ' such that the ordering of the filtration values is not impacted by changes of θ within this neighbourhood. Thus,

$$\exists \epsilon > 0 \forall h : |h| < \epsilon \implies y_i(\theta' + h) \neq y_j(\theta' + h)$$

and

$$y_{\pi(1)}(\theta' + h) < y_{\pi(2)}(\theta' + h) < \cdots < y_{\pi(n)}(\theta' + h).$$

This further implies that for a sufficiently small change of θ' the induced filtrations remain unchanged, i.e.

$$\left(K^{(i)}(\theta')\right)_{i=0}^{n} = \left(K^{(i)}(\theta' + h)\right)_{i=0}^{n}$$

Importantly, this means that the *selection* of persistence pairs $(f^{(i)}, f^{(j)}) \in \mathcal{D}$ will also remain the same (yet the filtration values associated with the tuple might differ). By applying the chain rule we can decompose the derivative

$$\frac{d\Psi(\mathbf{ph}(G, f_{\theta'}))}{d\theta} = \frac{d\Psi(\mathcal{D})}{d\mathcal{D}} \frac{d\,\mathbf{ph}(G, f_{\theta'})}{d\theta}$$

as Ψ is assumed to be differentiable, we solely need to show that the derivative of $\mathbf{ph}(G, f_{\theta'})$ exists:

$$\begin{aligned}
&\lim_{|h|\to 0} \frac{\mathbf{ph}(G, f_{\theta'}) - \mathbf{ph}(G, f_{\theta'+h})}{h} \\
&= \lim_{|h|\to 0} \frac{\{(f(\theta', v_{\pi(i)}), f(\theta', v_{\pi(j)}))\}^{i<j} - \{(f(\theta' + h, v_{\pi(i)}), f(\theta + h, v_{\pi(j)}))\}^{i<j}}{h} \\
&= \lim_{|h|\to 0} \frac{\{(f(\theta', v_{\pi(i)}) - f(\theta' + h, v_{\pi(i)}), f(\theta', v_{\pi(j)}) - f(\theta + h, v_{\pi(j)}))\}^{i<j}}{h} \\
&= \left\{\left(\lim_{|h|\to 0} \frac{f(\theta', v_{\pi(i)}) - f(\theta' + h, v_{\pi(i)})}{h}, \lim_{|h|\to 0} \frac{f(\theta', v_{\pi(j)}) - f(\theta' + h, v_{\pi(j)})}{h}\right)\right\}^{i<j} \\
&= \left\{\left(\frac{f(\theta', v_{\pi(i)})}{d\theta'}, \frac{f(\theta', v_{\pi(j)})}{d\theta'}\right)\right\}^{i<j},
\end{aligned}$$

where steps 2 and 3 are only possible because the matching remains consistent. □

We were thus able to show that we can propagate gradients through the persistent homology computation while only relying on the assumption that the filtration values are distinct. It should be noted here, that while this does allow the computation of a gradient, this gradient disregards any potential changes that could be induced by a change in the ordering and pairing of the filtration values. Recently, there has been a lot of development on *differentiable sorting* [21, 56], i.e. the inclusion of the sorting operation and its effects into the gradient computation. This represents a first step into the direction of differentiating through the persistent homology computation, yet further work on making the selection process differentiable is needed.

5.8.2 Expressive Power

As shown in the Section 5.4 the expressivity of GNNs has been extensively studied and most GNNs were found to be limited by the expressivity of the 1-WL test. In order to show that TOGL has *higher* expressivity than the 1-WL test, we will (1) that TOGL can differentiate all graphs that the 1-WL test can differentiate and (2) show how TOGL can trivially differentiate graphs that are indistinguishable by the 1-WL test . It is important to note, that the higher expressivity of TOGL does not imply that the model will generally show better performance, as in some cases being unable to distinguish different inputs can be an advantage. When CNNs are trained for image classification, the features are typically polled using average pooling such that the final representation does contain spatial information. This is beneficial in the case of image classification as the exact location of features is often irrelevant. Thus while the CNN shows lower expressivity in image classification this is actually beneficial as it creates an inductive bias compatible with the task. If we would train the same network to perform object discovery, this invariance and lower expressivity would on the other hand become a disadvantage. This is related to the well-known bias-variance trade-off in machine learning.

The computations of persistent homology are invariant to isomorphisms, thus in order to show the same expressivity as 1-WL it is sufficient for us to show that we can differentiate all graphs that the WL-test can differentiate. In particular, we show that there exists an injective filtration (in order to remain in line with Theorem 5.1), which when the label histograms of WL diverge would also lead to different persistent diagrams. We split the proof into multiple parts: (1) we show that persistent homology given an arbitrarily flexible filtration function can differentiate all graphs that the 1-WL test can differentiate, (2) we show that this filtration can be approximated to arbitrary precision using an injective function in order to ensure differentiability and (3) we show that utilizing this approximated function would still be able to differentiate graphs that a 1-WL test can differentiate.

Lemma 5.2 (Expressivity of Persistent Homology)**.** Persistent homology is at least as expressive at 1-WL, i.e. if the 1-WL label sequences of two graphs G and G' diverge, there is a filtration f such that the corresponding 0-dimensional persistence diagrams $\mathcal{D}_0$ and $\mathcal{D}'_0$ differ.

Proof. Assume that the label sequences of G and G' diverge at iteration h. Thus, $\phi_G^{(h)} \neq \phi_{G'}^{(h)}$ and there exists at least one label whose count is different. Let $\mathcal{L}^{(h)} := \{l_1, l_2, \dots\}$ be an enumeration of the finitely many hashed labels at iteration h. We can build a filtration function f by assigning a vertex v with label l_i to its index, i.e. $f(v) := i$, and setting $f(v, w) := \max\{f(v), f(w)\}$ for an edge (v, w). The resulting 0-dimensional persistence diagram $\mathcal{D}_0$ (and $\mathcal{D}_0'$ for G') will contain tuples of the form (i, j), and each vertex is guaranteed to give rise to *exactly* one such pair. Letting $\mu_0^{(i,j)}(\mathcal{D}_0)$ refer to the multiplicity of a tuple in $\mathcal{D}_0$, we know that, since the label count is different, there is *at least* one tuple (k, l) with $\mu_0^{(k,l)}(\mathcal{D}_0) \neq \mu_0^{(k,l)}(\mathcal{D}_0')$. Hence, $\mathcal{D}_0 \neq \mathcal{D}_0'$. □

We can understand this proof on a more intuitive level: Given that the label histograms of the 1-WL iterations differ, there will be a different number of vertices associated with a particular index i in two filtrations. As vertices with the same filtration value are created at the same step in the filtration, more or fewer points associated with a particular label would also lead to a different number of features to be created at a filtration step and thus lead to different multiplicities.

While Theorem 5.5 shows the existence of a filtration, it represents a constructive proof and relies on the existence of 1-WL labels. Further, the resulting filtration is not typically injective as the WL iterations might result in vertices being associated with the same label. We continue by showing that an injective filtration as applied in TOGL can approximate the above construction to arbitrary precision.

Lemma 5.3 (Approximation with Injective Function)**.** For all $\epsilon > 0$ and $f\colon V \to \mathbb{R}^d$ there exists and injective function $\tilde{f}$ such that $||f - \tilde{f}||_\infty < \epsilon$.

Proof. Let $V = \{v_1, \ldots, v_n\}$ the vertices of a graph and $\operatorname{im} f = \{u_1, \ldots, u_m\} \subset \mathbb{R}^d$ their images under f. Since f is not injective, we have $m < n$. We resolve non-injective vertex pairs iteratively. For $u \in \operatorname{im} f$, let $V' := \{v \in V \mid f(v) = u\}$. If V' only contains a single element, we do not have to do anything. Otherwise, for each $v' \in V'$, pick a new value from $\mathrm{B}_\epsilon(u) \setminus \operatorname{im} f$, where $\mathrm{B}_r(x) \subset \mathbb{R}^d$ refers to the open ball of radius r around a point x. Since we only ever remove a finite number of points, such a new value always exists, and we can modify $\operatorname{im} f$ accordingly. The number of vertex pairs for which f is non-injective decreases by at least one in every iteration, hence after a finite number of iterations, we have modified f to obtain $\tilde{f}$, an *injective* approximation to f. By always picking new values from balls of radius ϵ, we ensure that $\|f - \tilde{f}\|_\infty < \epsilon$, as required. □

Thus we can construct an injective function from a non-injective function by replacing non-injective input-output pairs with pairs where the output was perturbed within an ϵ-ball of the previous output value. By ensuring that the perturbed value is not already part of the function's image we can create an injective function. On $\mathbb{R}$ this is always possible as there are infinitely many points within an ϵ-neighbourhood of each point.

Finally, we need to show that Lemma 5.2 still holds despite $\tilde{f}$ being from the more restricted class of injective functions. For this, we show that the difference of the persistence diagrams for the filtration function f propagates into a difference in persistence diagrams for the approximated filtration function $\tilde{f}$.

Lemma 5.4. Let G and G' be two graphs whose 0-dimensional persistence diagrams $\mathcal{D}_0$ and $\mathcal{D}'_0$ are calculated using a filtration function f as described in Lemma 5.2. Moreover, given $\epsilon > 0$, let $\tilde{f}$ be an injective filtration function with $\|f - \tilde{f}\|_\infty$ and corresponding 0-dimensional persistence diagrams $\widetilde{\mathcal{D}_0}$ and $\widetilde{\mathcal{D}'_0}$. If $\mathcal{D}_0 \neq \mathcal{D}'_0$, we also have $\widetilde{\mathcal{D}_0} \neq \widetilde{\mathcal{D}'_0}$.

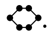
Proof. Since $\tilde{f}$ is injective, each tuple in $\widetilde{\mathcal{D}_0}$ and $\widetilde{\mathcal{D}'_0}$ has multiplicity 1. But under f, there were differences in multiplicity for at least one tuple (k, l). Hence, given $\tilde{f}$, there exists at least one tuple $(k, l) \in \widetilde{\mathcal{D}_0} \cup \widetilde{\mathcal{D}'_0}$ with $(k, l) \notin \widetilde{\mathcal{D}_0} \cap \widetilde{\mathcal{D}'_0}$. As a consequence, $\widetilde{\mathcal{D}_0} \neq \widetilde{\mathcal{D}'_0}$. □

Thus, given the previous steps we can now show that TOGL exhibits at least the expressivity of the 1-WL test:

Theorem 5.5 (Expressivity of TOGL)**.** Given a flexible class of injective filtration functions $\mathcal{F}$, persistent homology is at least as expressive as 1-WL, i.e. if the 1-WL label sequences for two graphs G and G' diverge, there exists a filtration $f \in \mathcal{F}$ such that the corresponding 0-dimensional persistence diagrams $\mathcal{D}_0$ and $\mathcal{D}'_0$ are not equal.

Proof. Take a non-injective filtration function f defined in Lemma 5.2 which results in $\mathcal{D}_0 \neq \mathcal{D}'_0$. According to Lemma 5.3, for every $\epsilon > 0$ we can find an injective function $\tilde{f}$ with $||f - \tilde{f}||_\infty < \epsilon$. Following Lemma 5.4, the persistence diagrams calculated by using this function as a filtration function are distinct, i.e. $\widetilde{\mathcal{D}_0} \neq \widetilde{\mathcal{D}'_0}$. Thus $\tilde{f}$ is a filtration function which differentiates two graphs when their 1-WL labels are distinct and is differentiable according to Theorem 5.1 □

The preceding theorem proves the *existence* of such a filtration function. Due to the capability of GNNs to approximate the Weisfeiler–Leman test [261] and the link between graph isomorphism testing and universal function approximation capabilities [50, Theorem 4], we can deduce that they are also able to approximate f and $\tilde{f}$, respectively. Yet, this does *not* mean that we always end up learning f or $\tilde{f}$. This result merely demonstrates that our layer can theoretically perform *at least as well as* WL[1]. In practice, TOGL may learn other filtration functions and injective filtrations based on 1-WL are not necessarily optimal for a specific task.

To prove that our layer is more expressive than a GCN, it is sufficient to show that there are pairs of graphs G, G' that 1WL cannot differentiate but that *can* be distinguished by persistent homology computations and thus by TOGL. Let G be a graph consisting of two triangles, i.e. , and let G' be a graph consisting of a hexagon, i.e. . 1-WL will be unable to distinguish these two graphs using degree-valued labels because all generated labels in all iterations will always be the same (see Figure 5.2 for a visualization of the WL label histograms on similar graphs). In contrast, persistent homology can simply distinguish the two graphs via their Betti numbers. For G we have $\beta_0(G) = \beta_1(G) = 2$ as G is composed of two connected components and two cycles, whereas G' has the Betti numbers $\beta_0(G') = \beta_1(G') = 1$ as it only contains a single cycle and a single connected component. Thus, the characteristics captured by persistent homology are elementary different from the ones captured by the 1-WL test. Combined with Theorem 5.5 this shows that persistent homology is strictly more powerful than the 1-WL when combined with sufficiently expressive filtrations. This further highlights the importance on being able to learn the filtrations such that this level of expressivity is (at least in theory) achievable.

5.8.3 Runtime

As shown in the introduction of this thesis, 0-dimensional persistent homology can be computed using a Union-Find data structure, which results in a runtime complexity of $\mathcal{O}(m\alpha(m) + m)$ where α denotes the slowly growing inverse Ackermann function and m the number of edges present in the graph. This process is repeated for each filtration f. Thus, compared to conventional GNNs TOGL adds $\mathcal{O}(n_f m\alpha(m) + n_f m)$ runtime complexity per included TOGL layer, where n_f corresponds to the number of filtrations used.

5.9 Empirical evaluation

We evaluate TOGL on a set of synthetic and real-world data sets, where our primary interest lies in understanding which scenarios benefit from the inclusion of topological features. For this create a set of purely structure-based data sets, where the vertex features are replaced with random values. Besides these artificially generated scenarios, we also evaluate TOGL on real-world graph benchmarks where it performs competitively to previous approaches.

5.9.1 Experimental Setup

We follow the setup of Dwivedi et al. [66] which proposes a consistent training setup across different datasets and models and ensures a fair comparison due to a limited parameter budget. In all figures and tables, we denote the mean test accuracy and standard deviation over different folds of the datasets. For datasets that don't have multiple folds defined we instead compute the performance over multiple training runs.

Comparison partners We compare to several GNN architectures, namely (1) Graph convolutional networks (GCN) [122], (2) Graph Attention Networks (GAT) [240], (3) Gated-GCN [30], (4) Graph Isomorphism Network (GIN) [261], (5) the Weisfeiler–Leman kernel (WL) [217] (6) and WL-OA [125] . These methods have been assessed in benchmarking papers [24, 66] and their setup is comparable to ours. In particular, we used the same folds and hyperparameters. It is worth noting though, that we assume the conventional stacking architecture commonly used in DNNs instead of the approach of concatenating the representations of all layers in order to derive a representation for the final graph or node classification layer (which for example is utilized in the original work of [261]). We ran most experiments ourselves to ensure a maximal degree of consistency and reproducibility.

TOGL MODELS In order to highlight TOGL s modularity and to compare how TOGL behaves when combined with different GNN architectures, we replace the second layer of all examined GNN models with a TOGL layer. This ensures, that the TOGL models contain approximately the same number of parameters and the same depth as their paired comparison partners. We insert TOGL specifically in the second position, as it allows harnessing the first GNN layer as a filtration function and still gives the later layers the opportunity to incorporate the topological information injected by TOGL.

5.9.2 Runtime

We empirically compared the runtime of TOGL with conventional graph neural networks by training both on the DD dataset. We found that the per epoch runtime of TOGL is 4.10 ± 0.17 seconds, whereas a same size GNN only requires 2.00 ± 0.12 seconds. Thus the empirical runtime of TOGL is almost factor 2 slower than a conventional GNN. While the theoretical overhead of computing persistent homology is almost negligible in the context of training neural networks, the practical implications of including TOGL are significant. This is mainly due to the lack of GPU-optimized implementations for the calculation of persistent homology that work well with deep learning frameworks. In our experiments, we implemented a CPU-based C++ implementation of the Union-Find algorithm in order to compute 0 and 1-dimensional persistent homology. This unfortunately leads to additional copy overhead between CPU and GPU which slow down training. Nevertheless, with the development of further optimizations and GPU-based algorithms, this gap is expected close to the negligible theoretical runtime difference.

5.9.3 Synthetic Data Sets

To first elucidate if our model can pick up the type of patterns we expect it to, we applied it to two datasets that cannot be differentiated using the 1-WL test but can be trivially differentiated via persistent homology and by the human eye. As an example, we replace the second layer of an n-layer GCN with TOGL (denoted as GCN-TOGL). The results of these experiments were presented in Figure 5.3 at the beginning of this section as a motivational example. Both binary classification datasets were generated with 1000 examples per class. In the CYCLES dataset (Figure 5.3, left),

Table 5.1: Results for the structure-based experiments. We depict the test accuracy obtained on various benchmark data sets when only considering structural information (i.e. the network has access to *uninformative* node features). Graph classification results are shown on the left, while node classification results are shown on the right. We compare three architectures (GCN-4, GIN-4, GAT-4) with corresponding models where one layer is replaced with a nd highlight the winner of each comparison in **bold**.

	Graph classification				*Node classification*
Method	DD	ENZYMES	MNIST	PROTEINS	PATTERN
GCN-4	68.0±3.6	22.0± 3.3	76.2± 0.5	68.8± 2.8	85.5 ± 0.4
GCN-3-TOGL-1	**75.1±2.1**	**30.3± 6.5**	**84.8± 0.4**	**73.8± 4.3**	**86.6 ± 0.1**
GIN-4	75.6±2.8	21.3± 6.5	83.4± 0.9	**74.6± 3.1**	84.8 ± 0.0
GIN-3-TOGL-1	**76.2±2.4**	**23.7± 6.9**	**84.4± 1.1**	73.9± 4.9	**86.7 ± 0.1**
GAT-4	63.3±3.7	21.7± 2.9	63.2±10.4	67.5± 2.6	**73.1 ± 1.9**
GAT-3-TOGL-1	**75.7±2.1**	**23.5± 6.1**	**77.2±10.5**	**72.4± 4.6**	59.6 ± 3.3

examples from one class are generated to be composed of a single large cycle, whereas the other class is composed of multiple small cycles with the same number of nodes in total. They can thus be easily differentiated by any approach which incorporates notions of topology as both the number of connected components (i.e. 0-dimensional topological features) and the number of cycles (i.e. 1-dimensional topological features) are different between the two classes. In the Necklaces dataset, we generate graphs of chains that either contain two separate cycles or a single merged one. Both classes have the same *number* of cycles, yet the incorporation of connectivity information along their neighborhoods makes them easily distinguishable for TOGL as filtrations to highlight this property can be learned easily.

As shown in Figure 5.2 the labeling scheme of the 1-WL test cannot differentiate these two structures easily and thus the WL graph kernel performs poorly in both tasks given a finite number of iterations. Interestingly, while the GCN performs poorly with a low number of layers its performance increases with increasing depth. The approaches which incorporate topology both perform better than their non-topological counterparts. While the approach based on static persistent homology with node degree filtration (PH) successfully differentiates the classes of the Cycles dataset due to their simple structure and the Betti numbers being discriminative in themselves, the performance is much lower on the Necklaces dataset. Further, the GCN performs significantly worse than GCN-TOGL in both scenarios, especially when composed of only a few layers.

5.9.4 Structure-Based Graph and Node Classification

As previously indicated (see Section 5.4.1), node features of graphs often already result in a very good performance [24]. Thus an evaluation with node features alone would not be sufficient to characterize the contribution of the *graph structure* to a model's performance. Motivated by these insights and the goal to highlight that TOGL allows a GNN to exploit topological information in the graph structure that would otherwise not be easily captured, we constructed graph datasets where the original node labels are replaced by random labels. This leaves only structural / topological information for the classification task. Solely in the case of the PATTERN dataset, we directly run evaluations on the original dataset as it is created by SBMs and the node features are random by design [66]. We do not include the Cluster dataset in this scenario as it represents a semi-supervised node classification task which is incompatible with randomizing the node labels.

Table 5.1 shows the results for both graph and node classification on such graphs. We can recognize that there is a *clear advantage* in using TOGL in most of the cases compared to the paired comparison partners. TOGL improves performance in almost all cases, in some cases, the performance increase is over 8% (see the comparison of GCN-4 and GCN-3-TOGL-1 on the MNIST dataset). Solely for the GIN model on Proteins and the GAT model on PATTERN we can observe a decrease in performance when including TOGL. It is important to note that the maximal performance a model can achieve on the structural MNIST dataset is at least 20% lower than for the non-structural case. This is due to the fact, that the node attributes contain the positional information of the superpixels. Without this information, it is impossible to for example differentiate a 9 from a 6 as the resulting graphs would be considered isomorphic.

To further show that TOGL allows GNNs to exploit topological information from the graph more easily and also at lower depth, we compared the performance of the considered models on the structural MNIST variant while changing the number of layers of the models. We present the results in Figure 5.5. From this experiment, it is evident that TOGL allows the model to access topological and structural features of the graph more easily and *at a lower depth*.

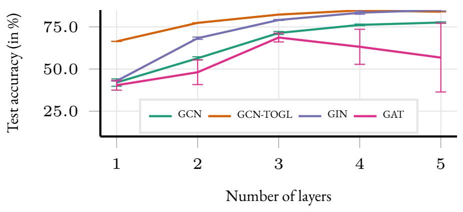


Figure 5.5: Classification performance when analysing the structural variant of MNIST.

5.9.5 Benchmarking Data Sets

After showing the utility of TOGL for tasks where topological and structural information is of crucial importance, we now move to the examination of TOGL's performance on standard graph and node classification tasks. We show the results of the evaluation in Table 5.2. We can recognize that the TOGL augmented models perform better than their paired comparison partners in most cases, which highlights the beneficial effect of substituting a conventional GNN layer with TOGL. Nevertheless, we observed strong performance degradation on the ENZYMES dataset

Table 5.2: Test accuracy on benchmark data sets (following standard practice, we report weighted accuracy on CLUSTER and PATTERN). Methods printed in black have been run in our setup, while methods printed in grey are cited from the literature, i.e. Dwivedi et al. [66], Morris et al. [164] for IMDB-B and REDDIT-B, and Borgwardt et al. [24] for WL/WL-OA results. Graph classification results are shown on the left, while node classification results are shown on the right. Following Table 5.1, we take existing architectures and replace their second layer with ; we use *italics* to denote the winner of each comparison. A **bold** value indicates the overall winner of a column, i.e. a data set.

	Graph classification							*Node classification*
METHOD	CIFAR-10	DD	ENZYMES	MNIST	PROTEINS-FULL	IMDB-B	REDDIT-B	CLUSTER
GATED-GCN-4	**67.3** ± **0.3**	72.9 ± 2.1	65.7 ± 4.9	**97.3** ± **0.1**	**76.4** ± **2.9**	—	—	60.4 ± 0.4
WL	—	77.7 ± 2.0	54.3 ± 0.9	—	73.1 ± 0.5	71.2 ± 0.5	78.0 ± 0.6	—
WL-OA	—	**77.8** ± **1.2**	58.9 ± 0.9	—	73.5 ± 0.9	74.0 ± 0.7	87.6 ± 0.3	—
GCN-4	54.2 ± 1.5	72.8 ± 4.1	**65.8** ± **4.6**	90.0 ± 0.3	*76.1* ± *2.4*	68.6 ± 4.9	**92.8** ± **1.7**	57.0 ± 0.9
GCN-3– 1	*61.7* ± *1.0*	*73.2* ± *4.7*	53.0 ± 9.2	95.5 ± 0.2	76.0 ± 3.9	*72.0* ± *2.3*	89.4 ± 2.2	*60.4* ± *0.2*
GIN-4	54.8 ± 1.4	70.8 ± 3.8	*50.0* ± *12.3*	*96.1* ± *0.3*	72.3 ± 3.3	72.8 ± 2.5	81.7 ± 6.9	58.5 ± 0.1
GIN-3– 1	*61.3* ± *0.4*	*75.2* ± *4.2*	43.8 ± 7.9	96.1 ± 0.1	*73.6* ± *4.8*	**74.2** ± **4.2**	89.7 ± 2.5	*60.4* ± *0.2*
GAT-4	57.4 ± 0.6	71.1 ± 3.1	26.8 ± 4.1	94.1 ± 0.3	71.3 ± 5.4	*73.2* ± *4.1*	44.2 ± 6.6	56.6 ± 0.4
GAT-3-TOGL-1	*63.9* ± *1.2*	*73.7* ± *2.9*	*51.5* ± *7.3*	*95.9* ± *0.3*	*75.2* ± *3.9*	70.8 ± 8.0	*89.5* ± *8.7*	*58.4* ± *3.7*

Table 5.3: Test accuracy when comparing TOGL (integrated into a simplified architecture) with existing topology-based embedding functions or `readout` functions. Results shown in grey are cited from the respective papers [42, 97]. For GFL, we cite degree-based results so that its performance values pertain to the same scenario.

METHOD	REDDIT-5K	IMDB-MULTI	NCI1	REDDIT-B	IMDB-B
GFL	55.7±2.1	49.7±2.9	71.2±2.1	90.2±2.8	**74.5±4.6**
PersLay	55.6	48.8	73.5	—	71.2
GCN-1-TOGL-1	**56.1±1.8**	**52.0±4.0**	**75.8±1.8**	90.1±0.8	74.3±3.6

which we attributed to overfitting. ENZYMES is the smallest dataset considered in this work and thus requires careful tuning of regularization hyperparameters in order to reach good performance. Yet, we did not explicitly tune the hyperparameters of our model to maintain comparability with previous large-scale GNN benchmarking studies [66]. Further, for GAT we observed a high degree of variance on the smaller scale datasets, yet note that this is in line with previous studies [66].

5.9.6 Comparison to Other Topology-Based Approaches

As TOGL relies on ideas such as differentiable filtrations and embedding functions which have been proposed by other approaches we take the time to explicitly compare these approaches to our proposed model. In particular, we compare to the approaches Graph Filtration Learning

(GFL) [97] and Persistence Layer (PersLay) [42]. The results of these comparisons can be found in Table 5.3 In many cases TOGL yields higher performance compared to the other topology-based approaches, the difference is most pronounced on the largest dataset NCI1 where TOGL shows higher performance by a margin of approx. 2.5%. Our approach thus performs better than using fixed filtrations as was suggested by Carrière et al. [42] and than incorporating the topological information only on the level of the `readout` function as suggested by Hofer et al. [97]. While the large standard deviations due to the rather small dataset sizes do not allow us to speak of significant differences, we hypothesize that the utilization of *multiple learned filtrations* (as done in our method) will yield more pronounced benefits on larger datasets.

5.10 Discussion and Conclusion

This chapter highlighted the problem of limited expressivity in GNNs and showed that these are often not able to differentiate graphs that can be trivially differentiated by humans when visualized in 2D. Further, it showed that in the evaluation of Graph Neural Networks the source of higher performance is often insufficiently investigated. For many approaches, it is *not* clear if they yield higher performance solely due to their reliance on more expressive models that can export hidden patterns in the node labels or if the models perform better as they can harness the information present in the graph in a better or more effective manner. In our work "Topological Graph Neural Networks", we strived to address these issues by (1) constructing a modular Graph Neural Network layer that can directly incorporate topological properties and structures of the graph into the computation of node representations and (2) evaluating our model in a manner that allows us to attribute the performance gains to the exploitation of the graph structure .

We showed that our method, TOGL, has *higher* theoretical expressivity than the 1-WL test for graph isomorphism, and therefore can increase the expressivity of GNNs by using persistent homology instead of by trying to climb the hierarchy of WL test expressiveness. We showed that on datasets with pronounced topological features, our method helps GNNs obtain substantial gains in performance. Further, we highlight that in the absence of node features our method makes it significantly easier for GNNs to exploit topological structures from input graphs leading to higher performance while requiring *fewer* GNN layers.

Our work also shows some limitations when examining the performance of models on graphs without node labels. Most approaches are not designed to work without node labels and thus should not be expected to perform well in this scenario. Further, our work showed that despite the higher theoretical expressiveness of TOGL, the model performs worse in some scenarios. There are multiple reasons why this could be the case. First, the discreteness of the persistent homology calculation and the lack of gradients with regard to the PH computation itself could lead to a higher degree of noise in the optimization process and could reduce performance. Moreover, for some problems, it might actually be beneficial if two types of graphs cannot be distinguished. Due to the potential false positives of the WL test for isomorphism, two graphs would be considered the same despite being different. This would introduce an *invariance* of the GNN to differences in graph structure that cannot be detected by the WL test. Similar to how convolutional neural networks are invariant to the positions of objects in images, this could actually represent a useful inductive bias. Nevertheless, it is worth noting that the WL test can differentiate most graphs such that such an invariance could only be of limited impact.

6 Evaluation of Graph Generative Models

While most of this thesis has concentrated on machine learning tasks, i.e. dimensionality reduction and classification, the following section will handle how to evaluate generative models, i.e. models which *generate* samples from a distribution that they were trained on. In some cases — in particular, when we are very familiar with the target structures — the evaluation and comparison of generative approaches can easily be done by eye. This, of course, impairs objective comparisons and the necessity of manual human evaluation prevents large-scale studies and makes results incomparable. The most prominent data falling into this category are generated natural images and faces. We humans can recognize mistakes in these data fairly well, although modern approaches are already partially capable of fooling us [170, 185, 226]. The recent development of machine learning methods has been driven by the iterative refinement on large benchmarking tasks with clearly defined evaluation criteria, such as the "ImageNet Large Scale Visual Recognition Challenge" (ILSVRC) [92, 126, 202]. Motivated by the progress that the ILSVRC challenge brought to the field, we want to accelerate the development of graph generative models by providing clear guidelines on how they should be evaluated in order to ensure the field moves in the right direction.

In the following, I will discuss how to evaluate generated structures that humans are not well able to differentiate due to our lack of experience with these structures. The chapter will be focusing on the evaluation of generated *graphs* and highlight issues with the current standard evaluation procedure in the domain. Nevertheless, many of the discussions in this section are not exclusively limited to graph generation but can also be applied to other types of data.

This section is partially based on the following work:
L. O'Bray*, M. Horn*, B. Rieck, et al. "Evaluation Metrics for Graph Generative Models: Problems, Pitfalls, and Practical Solutions". In: *International Conference on Learning Representations*. 2022

6.1 Graph Generative Models

Graph generative models are models that can generate graphs that follow the same distribution as graphs they have been trained on. The most prominent application of graph generative models is the generation and optimization of small molecules which can be represented as graphs, where nodes represent atoms and edges represent the connections between them. While graphs are discrete structures and thus cannot directly be optimized without requiring computationally costly discrete optimization, the latent representations of generative models are continuous and thus allow for optimization to take place. Thus an optimization algorithm can be used to traverse the space of graphs by traversing the latent space of a generative model trained on valid graphs [90].

A large variety of approaches have been developed to tackle this task relying on different approaches to generate graphs such as Variational Autoencoders [143, 221], Diffusion-based Generative Modelling [171], Generative Adversarial Networks [249], Autoregressive Modelling [139, 265] and Reinforcement Learning [264].

Compared to the generation of images, graphs are highly discrete structures (nodes or edges either exist or do not, there is no such thing as a partial edge), making the propagation of gradients difficult. This has been overcome in some of the above works by applying continuous relaxations to generated adjacency matrices and later applying a threshold to binarize the outcome [171, 221]. Others overcame this limitation by decomposing the generation process into a series of conditionally dependent steps where each step contains sampling from a discrete Bernoulli distribution [139, 265].

6.1.1 Challenges Evaluating Graph Generative Models

One of the core difficulties of evaluating graph generative models is that it essentially requires the comparison of two distributions *in graph space*. While approaches such as maximum mean discrepancy (MMD) allow for the comparison of two distributions using a kernel, the choice of this kernel for graphs is not trivial [25, 225].

In contrast, the progress of generative models in the image domain was initially based on visual inspection and later driven by metrics computed on the feature representation of trained deep neural network classifiers such as the Frèchet inception distance [95]. This style of evaluation is not available when working with graphs due to two reasons: First, humans don't have a good visual understanding of graphs as they are invariant to the reordering of nodes and thus their visualizations are invariant to any type of geometrical transformation. This lack of visual understanding is also due to insufficient experience with graphs: Graphs are not structures that we would be exposed to in the real world. While our brains have developed specialized circuitry to understand human faces, and natural scenes, this is not the case for graphs. Second, using a trained model to extract graph features for comparison is not possible due to the lack of a standardized dataset on which such a model would be trained. In the case of images, this was possible due to the existence of the ImageNet dataset [202] and the focus on real world images. Yet for graphs neither such a dataset nor the concept of "real world" is available.

Graphs are highly variable and complicated data structures, which require special care to be compared appropriately, this motivated our work to investigate how graphs can be compared in an unbiased manner and to see how the current standard of evaluating these models performs.

6.2 Comparing Graph Distributions

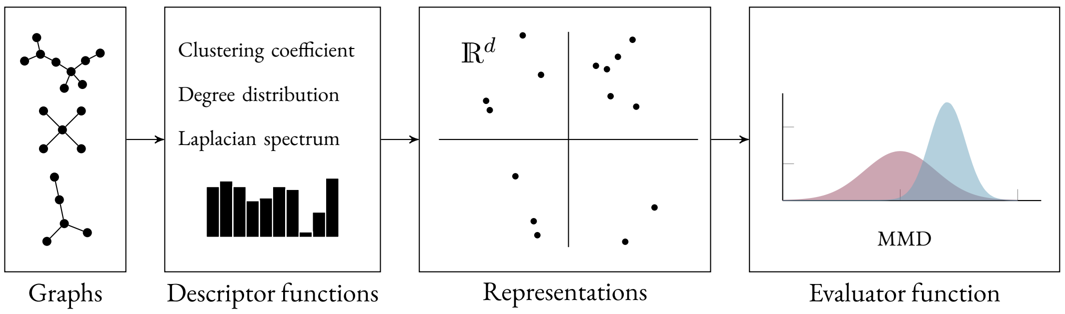


Figure 6.1: Overview of typical graph generative model evaluation workflow. Given a graph distribution, a set of graph descriptors is applied to each graph in order to map it to a high-dimensional representation in $\mathbb{R}^d$. These real-valued representations are used as a basis for the computation of an evaluator function, typically maximum mean discrepancy (MMD). While MMD does not rely on vectorial representations and can also directly operate on graphs using appropriate kernels [24], we find that its application to graph descriptors is much more common.

The following work concentrates on undirected graphs, which we denote using their vertices and edges as $G := (V, E)$. Further, we denote the original distribution of graphs (and thus the distribution that a generative model should try to mimic) as a multiset $\mathcal{G}^*$. When examining a set of models $\{\mathcal{M}_1, \mathcal{M}_2, \dots\}$ we examine the multisets of graphs generated by those models $[\mathcal{G}_1, \mathcal{G}_2, \dots]$. The goal of evaluating generative models on graphs in this context is to rank models based on their ability to mimic the data best, i.e. which distribution of generated data $\mathcal{G}$ is closest to the distribution of the testing data $\mathcal{G}^*$. To achieve this we require the definition of a (pseudo-)metric $d(\cdot, \mathcal{G}^*)$ to quantify the dissimilarity between any multiset of generated graphs and the original distribution. We argue that any such metric should be in line with the following desiderata:

1. **Expressivity:** When $\mathcal{G}^*$ and $\mathcal{G}$ do not originate from the same distribution, the metric should be able to detect this. In particular, $d(\mathcal{G}, \mathcal{G}^*)$ should monotonically increase with increasing as $\mathcal{G}$ and $\mathcal{G}^*$ become increasingly dissimilar.

2. **Robustness:** A suitable metric should be robust to small perturbations of the distribution $\mathcal{G}$. Ideally, the changes in the metric should be upper bound as a function of the degree of the perturbation. We prefer robust metrics as they are suited for handling the intrinsic randomness arising due to the training of deep neural networks.

3. **Efficiency:** The evaluation of the metric should be reasonably fast, to calculate. Despite evaluation being a posthoc analysis step, a suitable metric should scale well with increasing cardinality of the datasets and also with increasing size of the graphs themselves.

Computing distances on graph distributions is particularly difficult, as graphs have a varying number of nodes and edges and are only defined up to permutation. In fact, the task of simply determining whether two graphs are isomorphic, i.e. whether they represent the same graph in different permutations, is known to be of complexity NP, and the assignment of the *graph edit distance*, i.e. the number of editing operations needed to transform one graph into another graph is even NP-hard in general [268]. Thus while metrics like the graph edit distance might be the most appealing in terms of theory and correctness, they are precluded from our investigation as they contradict the efficiency requirement of our study as they do not allow efficient computation on arbitrary graph structures.

GRAPH DESCRIPTORS Instead, it is common practice to examine graph descriptors, which represent summary statistics of graphs that are invariant to permutations of the graph, i.e. isomorphic graphs would show the same graph descriptors. A graph descriptor f maps a graph G to some description of the graph in a descriptor space $\mathcal{Z}$ and reduces the problem of comparing graph distributions to the problem of comparing images of graph descriptor functions. Thus instead of computing the distance between the graph distributions $\mathcal{G}$ and $\mathcal{G}^*$ directly, we compare the distributions of graph descriptor functions, where $f(\mathcal{G}) = \{f(G_1), f(G_2), \ldots, f(g_n)\}$ and $f(\mathcal{G}) \subseteq \mathcal{Z}$. This allows the application of any statistical distance valid for data defined on $\mathcal{Z}$ and sidesteps the issue of defining these metrics directly on graphs. A visualization of this process is provided in Figure 6.1, here f can be any mapping that in some way characterizes the graph such as mapping the graph to a histogram of its degrees as a simple example.

It is important to note though, that the utilization of graph descriptor functions also leads to some disadvantages. In particular, the equality of graph descriptors between two graphs is a necessary but not sufficient condition for these to be isomorphic. While two isomorphic graphs would give rise to the same graph descriptor value, graphs might show the same graph descriptor value but not be isomorphic. Each graph descriptor value thus represents a perspective or projection of graphs into a space, where some information about the graph structure is lost. It follows that when using graph descriptors, it is important to consider multiple descriptors, as some might represent an overly simplistic view of the graph. A particularly strong example of such a simplistic descriptor function would be to use the number of nodes or edges in a graph, where all information about the connectivity structure is lost.

6.3 Current State of the Art for Graph Generative Model Comparison

Previous literature on the evaluation of Graph Generative Models has focused on one particular distance measure between graph descriptors in $\mathbb{R}^d$, the *maximum mean discrepancy* (MMD) [25, 88]. MMD enables the statistical comparison of two distributions using their means in a high-dimensional feature space implied by a kernel, without ever explicitly requiring to access the feature representation. This allows to compute the distance of two distributions using an infinite dimensional feature space representation (for example when using the RBF kernel) and further enables the computation of distances on structured objects if an appropriate kernels are defined. In the following, I will introduce MMD and showcase the descriptor functions typically used. The following introduction to MMD is inspired by a talk from Arthur Gretton at the Machine Learning Summer School 2020.

6.3.1 Maximum Mean Discrepancy

Maximum mean discrepancy is based on the idea that one can discriminate two distributions p and q by evaluating if their expectations with respect to all continuous bounded functions $C(\mathcal{X})$ always coincide [64, 88], i.e. $p = q$ if and only if $\mathbb{E}_{x\sim p}[f(x)] = \mathbb{E}_{y\sim q}[f(y)]$ for all $f \in C(\mathcal{X})$. Unfortunately, this is an extremely hard to compute condition such that in practice it is only evaluated with respect to a more limited class of functions $\mathcal{F}$. By instead phrasing the condition as a discrepancy that maximizes the difference of the expectations with respect to the function class $\mathcal{F}$, we can define a whole class of metrics[1] on probabilities, so-called *integral probability metrics (IPM)* which follow the form

$$\mathrm{D}[p, q, \mathcal{F}] := \sup_{f \in \mathcal{F}} \left(\mathop{\mathbb{E}}_{x\sim p}[f(x)] - \mathop{\mathbb{E}}_{y\sim q}[f(y)] \right), \tag{6.1}$$

where the function f that attains the supremum is often referred to as the *witness function*.

[1] Strictly speaking, these are not always metrics in a mathematical sense.

Maximum mean discrepancy is one particular IPM where the function class $\mathcal{F}$ is defined as the unit ball in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$. This norm is with respect to the RKHS norm of the function $f \in \mathcal{H}$ and thus represents a smoothness constraint [210], where the notion of smoothness is dependent on the kernel at hand and thus on RKHS induced by it. For a brief introduction to kernels and Hilbert spaces, we refer the reader to Section 3.1. The exact notion of smoothness relates to how kernels can be rewritten in terms of eigenvalues and eigenfunctions, which under certain constraints is possible for any positive semi-definite kernel according to Mercer's Theorem. Formally, the MMD is thus defined as

$$\mathrm{MMD}[p, q, \mathcal{F}] := \sup_{f \in \mathcal{H}, ||f||_{\mathcal{H}} \leq 1} \left(\mathop{\mathbb{E}}_{x \sim p}[f(x)] - \mathop{\mathbb{E}}_{y \sim q}[f(y)] \right). \tag{6.2}$$

Interestingly, this notion of MMD can be significantly simplified by utilizing some properties of the RKHS which will lead us to the practical definition of MMD on samples. In particular, for bounded reproducing kernels the expectation of a function applied to a random variable in the RKHS can be written as the inner product of the function's coefficients, with the expected features of the random variable, i.e. $\mathbb{E}_{x \sim p}[f(x)] = \langle \mu_x, f \rangle_{\mathcal{H}}$, where $\mu_x := \mathbb{E}_{x \sim p}[\phi(x)]$ represents the mean of the feature embedding ϕ of the random variable x in $\mathcal{H}$ [88]. While such a feature representation exists for any Hilbert Space, being able to rewrite expectations of functions as the inner product of their coefficients with mean embeddings is possible due to the reproducing property of the RKHS. It thus follows that the above statement can be written in terms of the differences in the expected feature vectors as

$$\begin{aligned}
\mathrm{MMD}[p, q, \mathcal{F}] &:= \sup_{f \in \mathcal{H}, ||f||_{\mathcal{H}} \leq 1} \left(\mathop{\mathbb{E}}_{x \sim p}[f(x)] - \mathop{\mathbb{E}}_{y \sim q}[f(y)] \right) \\
&= \sup_{f \in \mathcal{H}, ||f||_{\mathcal{H}} \leq 1} \left(\langle \mu_x, f \rangle_{\mathcal{H}} - \langle \mu_y, f \rangle_{\mathcal{H}} \right) \\
&= \sup_{f \in \mathcal{H}, ||f||_{\mathcal{H}} \leq 1} \langle f, \mu_x - \mu_y \rangle_{\mathcal{H}} \\
&= \left\langle \frac{\mu_x - \mu_y}{||\mu_x - \mu_y||}, \mu_x - \mu_y \right\rangle_{\mathcal{H}} \\
&= ||\mu_x - \mu_y||,
\end{aligned} \tag{6.3}$$

where in line three we utilize the fact that the inner product is maximal for vectors in the same orientation.

Finally, based on the previous definition of a kernel k (see Definition 3.1) for the RKHS $\mathcal{H}$, the squared MMD can then be written as

$$\begin{aligned}\mathrm{MMD}[p,q,\mathcal{H}]^2 &:= \|\mu_x - \mu_y\|^2 \\ &= \langle \mu_x - \mu_y \rangle_{\mathcal{H}} \langle \mu_x - \mu_y \rangle_{\mathcal{H}} \\ &= \langle \mu_x, \mu_x \rangle_{\mathcal{H}} - 2\langle \mu_x, \mu_y \rangle_{\mathcal{H}} + \langle \mu_y, \mu_y \rangle_{\mathcal{H}} \\ &= \mathop{\mathbb{E}}_{x\sim p, x'\sim p} k(x,x') - 2 \mathop{\mathbb{E}}_{x\sim p, y\sim q} k(x,y) + \mathop{\mathbb{E}}_{y\sim q, y'\sim q} k(y,y').\end{aligned} \tag{6.4}$$

A unbiased empirical estimate to Equation 6.4 is given as

$$\begin{aligned}\mathrm{MMD}^2[X,Y,\mathcal{F}] = &\frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j\neq i}^{m} k(x_i,x_j) + \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j\neq i}^{n} k(y_i,y_j) \\ &- \frac{2}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} k(x_i,y_j).\end{aligned} \tag{6.5}$$

6.3.2 Kernels and Graph Descriptor Functions

Table 6.1: The kernels & parameters chosen by three graph generative models for the MMD calculation.

Model		Kernel	Parameter choice σ and n_{bin}		
			Degree	Clustering	Laplacian
Model A	EMD	$\exp\left({}^{W(x,y)}/{}_{2\sigma^2}\right)$	$\sigma = 1, n_{\text{bin}} = d_{\text{max}}$	$\sigma = 0.1, n_{\text{bin}} = 100$	N/A
Model B	TV	$\exp\left(-\frac{\mathrm{d}_{\mathrm{TV}}(x,y)^2}{2\sigma^2}\right)$	$\sigma = 1, n_{\text{bin}} = d_{\text{max}}$	$\sigma = 0.1, n_{\text{bin}} = 100$	$\sigma = 1, n_{\text{bin}} = 200$
Model C	RBF	$\exp(-{}^{\|x-y\|^2}/{}_{2\sigma^2})$	$\sigma = 1, n_{\text{bin}} = d_{\text{max}}$	$\sigma = 0.1, n_{\text{bin}} = 100$	$\sigma = 1, n_{\text{bin}} = 200$

In the context of graph generative models, three kernels have been prominently used in combination with MMD to evaluate model performance. These include kernels based on the first Wasserstein distance (also referred to as earth mover's distance), based on the total variation (TV) distance, and the radial basis function kernel (RBF-kernel). Each of these kernels was used in a recent publication, which proposes its own generative model, making the metrics presented in the different publications incomparable. A summary of the kernels and the selected hyperparameters for different datasets is provided in Table 6.1.

In order to be able to use MMD with one of the above kernels, a featurization of the graphs into vectors in $\mathbb{R}^d$ is necessary as the above kernels are only defined for this case. We find that there are two types of descriptor functions used in literature, those relying on summary statistics of the graph such as the degree distribution histogram and the histogram of clustering coefficients and those relying on spectral properties such as the histogram of the Laplacian spectrum. Several works also rely on the orbit histogram as an additional graph descriptor. We refrain from its usage due to conflict with desideratum 3, which requires high scalability of the approach. We introduce some of the main descriptor functions below for ease of reference.

Degree distribution histogram Given a graph $G = (V, E)$ we can derive the degree histogram by computing the degree of each node, i.e by counting the number of neighbors each node has. A histogram with bin size 1 is then constructed using degrees by simply counting the number of times we observe each degree value. Thus the bin at index i corresponds to the number of vertices with degree i. By assuming a maximum degree d and setting the histogram bins on non-occurring degrees to zero we can thus construct a real-valued vector of fixed dimensionality, which is suitable for an MMD-based pipeline.

Clustering coefficient The (local) clustering coefficient can be computed on a per-vertex basis and is defined as the fraction of edges that exist among the neighbors of the node divided by the maximum possible number. The clustering coefficient $C(v) \in [0,1]$ thus determines how close the neighbors of a node are to a clique, i.e. a fully connected graph [250]. Formally, the clustering coefficient is defined as

$$C(v) := \frac{2|\{(v_i, v_j) \in E \mid v_i \text{ or } v_j \in \mathcal{N}(v)\}|}{\deg(v)(\deg(v) - 1)}, \tag{6.6}$$

where $\mathcal{N}(v)$ refers to the neighborhood and $\deg(v)$ to the degree of vertex v. We can convert the per-vertex clustering coefficients into a histogram and thus a vector of fixed dimensionality by deciding on the number of bins that should be used to split the range between 0 and 1 and thus converting the per-vertex values into a graph-level descriptor. It is worth noting though, that similar to the degree histogram this descriptor is inherently local and does not account for any global structure.

Laplacian Spectrum Histogram Spectral methods rely on the computation of spectra, i.e. of eigenvectors and eigenvalues associated with a matrix representation of the graph. Let $\mathbf{A}$ be the adjacency matrix of the graph, where $\mathbf{A}_{i,j} = 1$ if there is an edge between the vertices v_i and v_j, i.e. $(v_i, v_j) \in E$. The normalized graph Laplacian is then defined as $\mathcal{L} := \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{I}$ is the identity matrix and $\mathbf{D}$ represents the degree matrix, a diagonal matrix where the entries correspond to the degrees of the individual vertices, i.e. $\mathbf{D}_{i,i} = \deg(v_i)$. As the matrix $\mathcal{L}$ is real-valued and symmetric, it can be diagonalized and give rise to a full set of eigenvalues and orthogonal eigenvectors. The eigenvalues of $\mathcal{L}$ are positive and bounded, i.e. $0 \leq \lambda_i \leq 2$ [52], such that they can easily be summarized using a histogram with a fixed number of bins. While it is not totally clear how expressive this representation is as it is unknown whether the spectrum can fully determine the graph it was computed on [57], the Laplacian spectrum remains a prominent approach for deriving graph representations.

6.4 Issues with Current Practice

The common approach for evaluating graph generative models is to select hyperparameters for the kernels and descriptor functions and utilize them in combination with MMD in order to rank models according to their discrepancy to the true distribution. Here authors select different hyperparameters for kernel and descriptor functions, and to the best of our knowledge these values are predetermined, i.e. no selection is done to determine the adequate hyperparameters. While this is not a problem if the resulting rankings of models provided by MMD are stable with regard to the afore mention parameters, we find that in practice this is *not* the case and that the resulting rankings are highly sensitive. It goes so far that with a targeted selection of the hyperparameters, the rankings of models can be changed *almost arbitrarily!*

In the following sections, we highlight this problem using three recently developed graph generative models, GraphRNN [265] which constructs graphs using a deep autoregressive model in the form of a recurrent neural network, Graph Attention Network (GRAN) [139] which uses an attention-based hierarchical generation procedure, and Graph Score Matching [171] which utilizes score matching approaches to transform noise into graph structures. We trained each model on a series of synthetically generated graphs following the Community, Barabási-Albert, Erdös-Rényi, and Watts-Strogratz graph distributions and used the trained models to generate novel graphs from the datasets. We then calculated the MMD distance between the test graphs and the generated graphs of the models while varying (1) the kernel used (EMD, TV, and RBF), (2) the kernel hyperparameters $\sigma, \lambda \in \{10^{-5}, \dots, 10^{5}\}$ and (3) the descriptor function (degree histogram, clustering coefficient histogram and Laplacian spectrum histogram). In order to simplify matters, we solely refer to the hyperparameter of the kernel as σ independent of the kernel used. Further, as we solely want to highlight the presence of a problem and not to comment on the performance of individual approaches, we refer to the models as "A", "B", and "C".

6.4.1 Using MMD for the Evaluation of Graph Graph Generative Models

While MMD represents the current state of evaluating Graph Generative Models, it is important to note that its original application domain is the construction of two-sample tests. It was thus not directly designed for the comparison of graph generative models and has not been thoroughly benchmarked in this context. Nevertheless, MMD has now been utilized in a variety of different contexts such as the training of generative models [67, 138], and has recently also been shown to be beneficial in the context of evaluating generative models when components of MMD are optimized to maximize the power of a two-sample test [225].

The main issue at hand in the context of evaluating graph generative models is the reliance on graph descriptor functions prior to the application of MMD. While MMD represents a metric when a *characteristic kernel*[2] is used and thus $\mathrm{MMD}[p, q] = 0$ if and only if $p = q$, we must keep in mind that the MMD is actually evaluated on the graph descriptors. When using a characteristic kernel we can thus only make statements about the distribution of the graph descriptors and not about the graphs themselves. We highlight these issues in more detail subsequently.

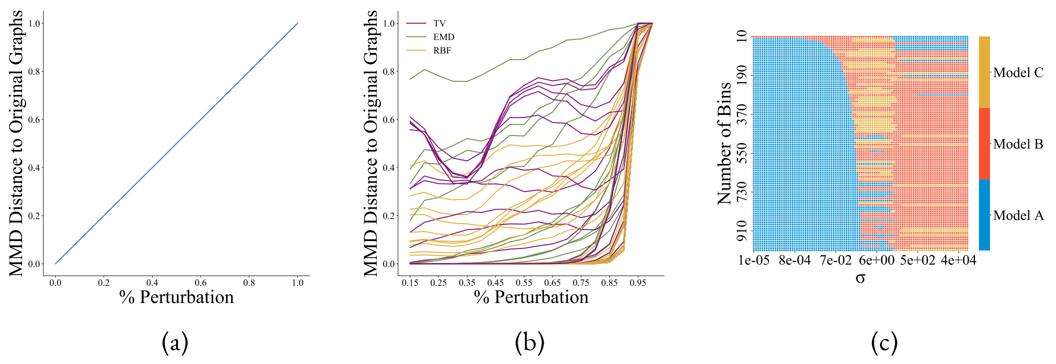


Figure 6.2: MMD-based evaluation with graph descriptors (b) is not in line with the expected ideal behavior (a) and leads to arbitrary rankings of models (c). a) A depiction of how we would want an evaluation metric to ideally behave when evaluated on a perturbed dataset. b) The behavior or MMD applied to the clustering coefficient graph descriptor as the degree of perturbation is increased. Each line corresponds to a unique kernel and hyperparameter combination. The perturbation randomly introduces edges into the graph. c) Model with the lowest MMD evaluated for different hyperparameters on the clustering coefficient graph descriptor using an RBF-kernel. Each square in the plot shows which of the models considered is ranked best given selections of the two possible hyperparameters of length scale and the number of bins used in the featurization.

Creating an external sense of dissimilarity via graph perturbations As two distributions become increasingly dissimilar we assume that their distance should monotonically increase as a function of their dissimilarity. An example of this behavior is shown in Figure 6.2a. As we are trying to estimate dissimilarity between distributions in the first place it is hard to evaluate if this is generally the case without relying on an external notion of dissimilarity. In our experiments, we introduce this external notion by applying perturbations to the originally generated data sets and utilizing the degree of perturbation as a proxy for how dissimilar we expect the two distributions to be.

[2] A kernel for which the mean feature map is injective.

We describe all graph perturbations used based on an example of a single graph $G := (V, E)$ where V refers to the vertices of the graph and E to the edges.

Add Edges

For each $v_i, v_j \in V$ with $v_i \neq v_j$ a sample from a Bernoulli distribution $x_{ij} \sim \text{Ber}(p_{\text{add}})$ is drawn. Samples for which $x_{ij} = 1$, are added to the list of edges such that $E' = E \cup \{(v_i, v_j) \mid x_{ij} = 1\}$.

Remove Edges

For each $e_i \in E$ a sample from a Bernoulli distribution $x_i \sim \text{Ber}(p_{\text{remove}})$ is draw, samples with $x_i = 1$ are removed from the edge list, such that $E' = E \cap \{e_i \mid x_i \neq 1\}$.

Rewire Edges

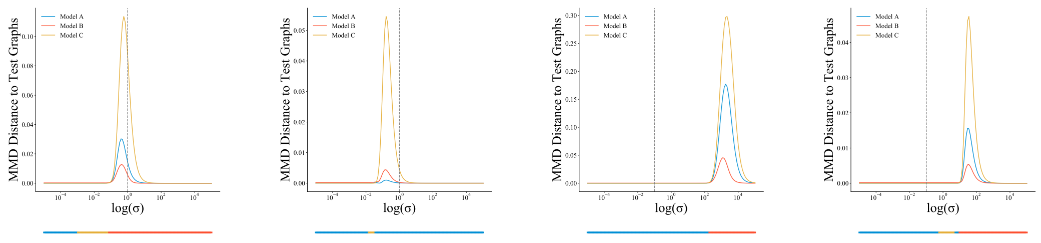
For each $e_i \in E$ a sample from a Bernoulli distribution $x_i \sim \text{Ber}(p_{\text{rewire}})$ is drawn, samples with $x_i = 1$ are rewired. For rewiring a further random variable $y_i \sim \text{Ber}(0.5)$ is drawn which determines which node $e_i[y_i]$ of the edge e_i is kept. The node to which the edge is connected is chosen uniformly from the set of vertices $v \in V$, where $v \notin e_i$, i.e. avoiding self-loops and reconnecting the original edge. Finally the original edge is removed and the new edge $e'_i = (e_i[y_i], v)$ is added to the graph $E' = E \cap \{e_i \mid x_i \neq 1\} \cup \{e'_i \mid x_i = 1\}$.

Add Connected Nodes

We define a set of vertices to be added $V^* = \{v_i \mid |V| < i \leq |V| + n\}$, where n represents the number of nodes to add. For each $v_i \in V$ and $v_j \in V^*$ we draw a sample from a Bernoulli distribution $x_{ij} \sim \text{Ber}(p_{\text{connect\_node}})$ and add an edge between v_i and v_j to the graph if $x_{ij} = 1$. Thus $E' = E \cup \{(v_i, v_j) \mid v_i \in V, v_j \in V^*, x_{ij} = 1\}$.

For each of the above perturbations, we can vary the degree of the perturbation by changing the parameters of the perturbation. All perturbations contain a set of parameters at which the distribution of graphs remains unchanged, which we set as the lowest perturbation value.

MMD DOES NOT ALWAYS CAPTURE DIFFERENCES IN DISTRIBUTION While it is possible to explicitly construct cases where the degree of perturbation will not relate to the dissimilarity , it highlights that the choice of descriptor function is the main determinant for which types of distributional changes can be detected or not. Generally, MMD will not increase monotonically as the two graph distributions become increasingly dissimilar as shown in Figure 6.2b while relying on the same descriptor function. This highlights that even when the graph descriptor is assumed to be predetermined, the selection of the MMD kernel and the kernel hyperparameters has a strong impact on the results and can lead to different conclusions. We apply perturbations to the initially generated graphs and compute the MMD while increasing the degree of perturbation and thus the



(a) EMD, degree (b) RBF, degree (c) EMD, clustering coef. (d) RBF, clustering coef.

Figure 6.3: MMD distance of generated graphs by three recent graph generative models to the test set of community graphs dataset. Each subfigure shows the values of MMD while varying the value of σ, the bar under the graph shows the best-ranked model according to the MMD distance. The grey line indicates the value of σ chosen by the authors. Subfigures 6.3a and 6.3b highlight the impact of switching between the EMD and the RBF kernel and show that this can lead to different conclusions when σ is selected to remain the same. Subfigures 6.3c and 6.3d show how the selected hyperparameter choices of previous publications miss the area of highest discriminative ability of MMD.

distance of the perturbed distribution to the original data distribution. Despite the distributions becoming increasingly dissimilar by design, a large number of kernel and hyperparameter combinations *fail* to capture this difference as is shown in Figure 6.2b. Often the distance remains close to constant even though the degree of perturbation is increased significantly or the distance might even decrease as the perturbation is increased. Finally, MMD values as a function of the degree of perturbation seem extremely sensitive to the choice of hyperparameters, leading to a large variety of different curve shapes.

MMD HAS NO INHERENT SCALE A further issue, especially regarding the comparability of different studies, is the fact that there is no inherent scale for MMD and the scale of values is highly specific to the kernel. It is thus difficult to assess if differences in the MMD value reflect *substantial* differences in the ability of models to mimic the distribution they were trained on. For example, it is not clear if a difference of 10^{-7} should be considered meaningful. Generally, many hyperparameters selected for evaluating models using MMD should be considered suboptimal with respect to the discriminative ability of MMD [225], such that studies might be able to obtain arbitrarily low MMD values. A visualization of this issue is provided in Figure 6.3. Here we see that depending on the kernel and the selected lengthscale of the kernel, MMD values live on extremely different scales and thus are not always comparable. This makes sense if we revisit the notion of MMD defined as the difference between mean embeddings in the RKHS (see Equation 6.3). These embeddings are highly dependent on the kernel and its hyperparameters and thus the differences between the embeddings would also vary greatly. Overall, this problem is especially critical as MMD results are typically simply reported in a table, such that the lack of inherent scale makes them extremely difficult for the reader to interpret. This is somewhat similar to presenting the accuracy of a model on a dataset where the class imbalance is not known. If the reader cannot estimate what the performance of a random classifier using only the class distribution would be, it is impossible for him to put the reported performance into context.

6.4.2 Implications of Kernel Selection

The selection of the kernel function has many implications for rendering specific evaluation procedures feasible or interpretable. For example, the kernel can impact the scalability of the approach if its runtime is superlinear, i.e. if the runtime of the kernel evaluated on a single pair of examples scales more than linearly as this effect compounds with the scaling of MMD which requires an evaluation on all pairwise examples[3]. This is for example the case with the earth mover's distance kernel, which requires solving an optimal transport problem for each pair of graphs and thus suffers from poor scalability [139].

A further issue is the fact that previous work partially relied on kernel functions which are not actually positive semi-definite. The most prominent example is the computation of the total variation distance on histograms, which leads to an *indefinite* kernel [174]. In this case, it is not clear if MMD applied to datasets using this kernel is actually meaningful as the behavior has not yet been studied and the derivation of MMD would also be partially invalid as it relies on the properties of the RKHS which are induced by positive definite kernels.

Finally, the most striking issue arises in the ranking of models, where a change of the kernel can result in a different ranking of the models being evaluated. An example of this occurring is given in Figure 6.3a and Figure 6.3b, where changing the kernel results in a different ranking of the methods, while keeping the lengthscale parameter of the kernel constant. This behavior is highly undesirable and problematic as it allows depicting results from very different perspectives by selecting the kernel appropriately and thus can lead to unfair comparisons where the reader might be misguided to believe that a method consistently outperforms another approach.

6.4.3 Implications of Hyperparameter Selection

While the selection of a kernel can be considered an *a priori* choice in the experimental design, often no clear process is followed to select the hyperparameters of the kernel to ensure high discriminative ability of MMD. This is especially problematic as similar issues can arise with the selection of the kernel itself because both have implications on the feature representation in the Hilbert space and thus can lead to the MMD values being on different scales and the rankings of models

[3]While there are more scalable approximation to MMD, we do not consider them in this work

to potentially change. Nevertheless, it is surprising *to what extent* the selection of hyperparameters already has a detrimental effect on the comparability of MMD results and their interpretability.

As shown in Figure 6.3c, the hyperparameters of the kernel have a strong impact on the scale of the MMD values, and selecting larger or smaller values for the lengthscale can influence which of the models is considered best. Further, it is important to note, that the parameters used in previous literature (dotted line in the figure) do not coincide with the range in which MMD has its largest discriminative ability (which would be at the point where the estimated values from MMD become maximal, thus in the range of 10^1 to 10^3). Regions of little change in MMD value, are generally considered suboptimal choices for high discriminative ability of MMD [225], and selecting hyperparameters in the range of maximal MMD values is recommended by recent literature [79]. Finally, this of course highlights that the hyperparameters of the kernel should be decided on a *per-kernel basis* as the areas of maximal MMD values are not consistent across different kernel choices contrary to common practice where a single value is decided upon *a priori*.

Similar to the selection of kernels, the selection of kernel hyperparameters can also influence the ranking of models. In Figure 6.2c we can see that this effect occurs both for the kernel hyperparameters as well as for parameters that could be considered part of the graph descriptor function (in this case the number of bins used in the histogram). Here we can see that while in many cases there seems to be a trend of which model shows lower MMD values (most evident in Figure 6.3), the actual ranking changes dependent on the hyperparameter selection. This goes so far that in Figure 6.2c we could rank *any* model as the best dependent solely on the selection of hyperparameters.

6.5 Guidance on Using MMD for the Evaluation of Graph Generative Models

In order to sidestep the potential pitfalls detailed above we now provide a set of suggestions on how to better leverage MMD in the evaluation of graph generative models. These include suggestions that allow readers of papers to better estimate the performance of models and hopefully will give some guidance to the field in general with regard to moving forward.

Provide a sense of scale The easiest and probably most powerful step to enhance the interpretability of results based on MMD is to provide a sense of scale. This allows readers to understand which MMD values should be considered “good”. A simple way this can be accomplished is by additionally reporting the values of MMD between the training and testing data. This will give additional information and allow to understand what the performance of two indistinguishable datasets would look like. Further, negative examples of dissimilar datasets can be provided by using graph perturbations.

Use valid and efficient kernels We recommend the usage of efficient kernels, which allow results to be reproduced with relatively little effort and encourage the usage of kernels that are positive semi-definite. In particular, we discourage the use of the total variation kernel as its effect on the applicability of MMD for the comparison of distributions is not fully understood. Instead, we recommend relying on either the RBF/Gaussian-kernel or the exponential/Laplace-kernel as they both are well characterized in the context of MMD and represent *universal* kernels which ensure that $\text{MMD}[p, q] = 0$ only when $p = q$ and thus define valid metrics on the distribution of graph descriptors.

Use meaningful descriptor functions Different descriptor functions measure different aspects of graphs and can thus be highly dependent on the task at hand. We generally recommend the utilization of previously suggested graph descriptor functions in the form of the degree histogram, the clustering coefficient histogram, and the Laplacian spectrum histogram and recommend making further adaptions if necessary for the application domain. A further recently developed alternative is the utilization of random features based on untrained graph neural networks [229]. While this line of research is still in its early steps it might provide a promising avenue to avoid handcrafted graph descriptor functions in the future.

6.5.1 Strategies for Selecting Kernels and Hyperparameters

In order to ensure that MMD is set up to recognize differences in the graph distribution appropriately we recommend starting from a controlled scenario. This is necessary as, in contrast to images, graphs cannot be easily compared by eye and thus we cannot rely on literal eye measurements for guidance. By starting with a controlled scenario where the practitioner knows that the graphs follow different distributions, he can ensure that the values of MMD are selected adequately to *at least* cover the known differences. We recommend doing this by applying random perturbations to the graphs of the dataset of interest, as these can be applied to *any dataset* and thus this routine can be done also in scenarios where there is no way to influence the true data-generating process as is the case for real-world graph datasets.

In the ideal case, we would want an evaluation metric where the degree of perturbation of a graph is adequately reflected, i.e. that the distance metric increases monotonically as a function of the perturbation intensity. By using perturbations, we can simultaneously assess the expressivity and sensitivity of a set of kernel and hyperparameters (or of any evaluation metric for generated graphs general) as we would want the metric to increase as the degree of perturbation increases (capturing its expressivity), but also that small changes in the degree of perturbation should only lead to small changes in the metric (capturing its robustness). In the following, we present an example of how such an analysis could be performed.

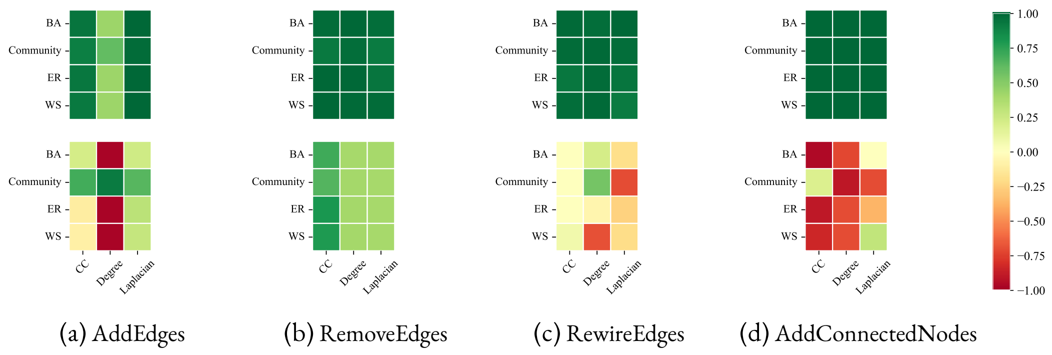


Figure 6.4: Correlations of MMD with the degree of perturbation of graphs compared for different descriptor functions and datasets (BA: Barabási-Albert Graphs, ER: Erdös-Rényi Graphs, WS: Watts-Strogatz Graphs). Given our expectations of an ideal metric, we would expect a correlation close to 1. The upper row shows the best combination of kernel and hyperparameters, i.e. with the highest correlation to the perturbation degree. The lower row shows the worst combination of kernel and hyperparameters measured in the same terms. It is evident that the selection of appropriate kernel and hyperparameters leads to a strong correlation with the perturbation whereas the converse is also possible for inappropriate kernel and hyperparameter combinations.

Correlation analysis to choose kernel and hyperparameters Given our previous requirements for a good metric, we now explore how to select a set of parameters that best align with these requirements. For this, we perturb graphs to varying degrees and measure the Pearson correlation between the degree of perturbation and the MMD computed using a specific combination of kernel and hyperparameters. While the Pearson correlation is a measure of linear correlation, and the relation between a perturbation and its effect on the metric might not be linear and still fulfill our requirements, we selected it due to its straightforward interpretability. We find that we can *always* find combinations of kernels and hyperparameters which lead to strong correlations of the MMD-derived metric with the degree of perturbation of the graphs. Besides reporting the best correlations achievable for each dataset and perturbation, we also report the worst possible combinations as this highlights the worst-case scenario of inappropriate selection of kernel and hyperparameters due to the lack of agreed-upon selection procedures. The results of the correlation analysis are shown in Figure 6.4. While in the top row we can see that robust combinations of kernel and hyperparameters exist which well reflect the degree of perturbation, the bottom row shows that in many cases the MMD does not show any correlation with the perturbation and might even show a negative correlation if the parameters are selected incorrectly.

To choose a kernel and hyperparameter combination we suggest picking the setting which shows the largest correlation with a perturbation that is meaningful in the application domain or if no such perturbation is known to select the hyperparameters base on the average correlation across different perturbations.

6.6 Conclusion and Discussion

In this chapter, I presented a detailed exploration of the pitfalls that occur when evaluating graph generative models using MMD. Our work shows that the selection of kernel, kernel hyperparameters, and graph descriptor functions are all of great importance for a representative evaluation of graph generative models. We highlight that if these parameters are not adequately selected, the evaluation via MMD becomes essentially meaningless and allows the generation of arbitrary ranking of models. To mitigate these problems we propose to practitioners the examination of the discriminative performance of MMD on perturbed versions of the datasets they are working on. This allows us to understand which parameters would represent distances in the space of graph distributions realistically and thus provide important guidance in their selection. Further, we recommend practitioners to provide MMD metrics with additional information to understand which values would correspond to very close and very distant distributions.

Our exploration has some limitations though. In particular, we rely on graph perturbations in order to create a continuum of increasingly distinct graphs. While this makes us invariant to the actual graph generative process and allows an application to real-world datasets, it assumes that graphs become increasingly dissimilar with increasing perturbation strength. For example, if the training dataset contains Erdős–Rényi graphs, the randomly rewiring edges (independent of how frequently) should not have a significant impact on the distribution of graphs. Further, the graph perturbations only represent axes along which we can characterize the behavior of evaluation metrics, but do not sample the complete space of negative samples. This could to some degree be avoided by not only applying one perturbation but combining different perturbations of different strengths. Additionally, our work is purely empirical and a more theoretical treatment of the pitfalls of current evaluation strategies would be warranted. One approach might be the examination of other graph generative procedures (i.e. configuration models) that are guaranteed to give rise to certain properties in the overall graph distribution. This might allow relating the ability of current evaluation strategies to quantify differences in the graph distribution with the differences in the underlying properties of the generative process or the derivation of bounds that relate the degree of perturbation with the amount of data needed for its detection.

A further issue resides in the fact that our work does not provide a direct solution to the selection of graph descriptors and MMD hyperparameters but solely provides guidance. If for instance, the graph descriptor functions are not appropriately selected, our advice on MMD hyperparameter selection will also not be beneficial. The reliance on a graph descriptor function is generally, problematic as it determines the perspective that MMD has on the graph. Here the exploration of graph kernels [24] for the evaluation of graph generative models could be a valuable line of future research. Concurrent work has suggested ways to avoid the requirement of a graph descriptor function, by relying on random features extracted from a graph neural network and comparing those using an RBF kernel [229]. We think this represents a promising perspective for future work and assume that the utilization of random features could be synergistic with the advice we present in our work.

7 Summary and Outlook

This chapter briefly summarizes the contributions of this thesis, discusses implications, and provides an outlook over potential future work. It is intended to provide a more holistic perspective compared to the discussions of the individual chapters and point to new research opportunities that arise from the work presented in this thesis.

* * *

The first part of the thesis focused on the problem of dimensionality reduction and how the power of non-linear deep neural networks can be harnessed to derive meaningful lower-dimensional representations. In Chapter 2, I present such a method by combining autoencoders with topological signatures computed via persistent homology. It represents the first method to leverage persistent homology in order to preserve topological structure across spaces. In contrast to existing dimensionality reduction approaches, our method handles multi-scale relationships in the data uniquely well. It thus represents a promising candidate for applications in the life sciences such as the exploratory analysis of high-dimensional single-cell sequencing data. The approach relies on the propagation of gradients through the persistent homology computation and thus shows similar limitations to Chapter 5 with respect to the discrete persistent homology computation. A future research direction is thus the exploration of *soft* persistent homology, where birth and death times could be extended to carry probability weights. The resulting distributions over persistence diagrams could lead to potentially more informative gradients in the downstream processing. It could thus improve all methods in this thesis that rely on the computation of gradients through the persistent homology computation. While evolutionary strategies represent a promising direction in that regard (for a discussion see Chapter 2), the exploration of point processes for the description of persistence diagrams also represents an interesting avenue.

* * *

The second part of my thesis presented methods tackling irregular sampling of medical data and discussed problems related to their evaluation. In Chapter 3, I showcased a new preprocessing approach for MALDI-TOF spectra and a kernel for machine learning on MALDI-TOF spectrum peaks. Our work represents the first kernel designed for machine learning on MALDI-TOF spectra and shows much higher performance than commonly applied machine learning methods in a microbial resistance prediction task. The presented kernel accounts for the fact that minor shifts in peak positions of MALDI-TOF spectra can be due to technical variability. By combining the kernel with a Gaussian process classifier we additionally showcase the ability to reject out-of-distribution samples, which makes our approach especially well-suited for clinical applications. Future research directions include the incorporation of additional inductive biases in the form of charge-multiple invariance and the exploration of domain adaptation approaches to counteract hospital-specific device settings and lab preparations that introduce distribution shifts.

In Section 4.2 I present an alternative approach for classifying irregularly-sampled data based on learning set functions with deep neural networks. I demonstrate the viability of this approach on medical time series data, where it can be uniquely applied without prior binning or imputation. The work shows that while the model performs competitively with recent approaches, it does not lead to state-of-the-art performance in medical times series tasks. Here the incorporation of translation equivariance represents a promising direction to improve performance and increase robustness in future work. A further avenue of future research represents the exploration of similar set-function-based approaches to MALDI-TOF data while incorporating as much data-specific inductive biases as possible. While this might be successful on larger quantities of data, until recently [251], large-scale MALDI-TOF data has only scarcely been publicly available.

In Section 4.3, I presented an evaluation of medical time series models with a focus on their capability to transfer to other datasets in the context of a sepsis early prediction task. Here, I showed that deep learning approaches can achieve very high sepsis detection performance at a high recall level. Further, I showed that contrary to intuition, deep learning models showed higher performance compared to less expressive models when being transferred to other datasets. The study presents the first homogenized multi-center ICU cohort for sepsis early prediction and thus a solid foundation for future work. Potential follow-up research includes the exploration of domain-adaptation approaches to leverage the multiple data sources and increase generalization to new unseen data and trying to avoid indication bias by imputing the data prior to model application.

Taking a holistic perspective on the issues discussed in Chapter 4, there are synergies that can be exploited between different solution approaches. In particular, the irregularly-sampled nature of medical times series and indication biases are very related. Generally, being able to exploit sampling information is important for model performance in a naive evaluation scenario (such as measuring the predictive accuracy on i.i.d. data). Nevertheless, it can actually *reduce* generalization performance due to distribution shifts when treatment procedures differ as even the patterns in which variables are measured might have an impact on model performance. These issues can be further amplified by feedback loops due to the interaction of the model and clinician due to increased reliance on model predictions. When the models' predictions start to influence the clinician, this can introduce a distribution shift relative to what the model was trained on. One approach to avoid this collapse could be the construction of counterfactual patient models, which could be used to obtain deconfounded data [189, 212]. A clinical support system could then be trained on both the real and the deconfounded data to ensure that it does not overly rely on the interventions from clinicians to predict patient outcomes.

* * *

In the third part of my thesis, I presented a method for more expressive machine learning on graphs and which problems arise when evaluating generated graphs. In Chapter 5, I showed how to improve supervised learning on graphs by extending graph neural networks to have access to multi-scale topological information of the graph. The presented approach outperforms baselines in particular in the absence of node features. In contrast to previous topologically inspired approaches, our method is highly modular, does not require the pre-computation of features to augment the graph, and can be combined with any standard graph neural network architecture. Future research directions are similar to those presented in Chapter 2 and focus on improving gradients through the discrete persistent homology computation. Further, a detailed investigation of the trade-offs between expressivity and invariances in graphs represents a promising avenue for future research and would be of great utility to the community.

In Chapter 6, I showed that the most common approach for evaluating generated graphs can lead to incorrect results when not applied carefully. I highlighted that the adversarial selection of hyperparameters of kernels and graph descriptors can lead to arbitrary rankings of model performances. Further, I show which combinations are more robust by comparing different settings under graph perturbations. Graph generative modeling is still a nascent field and due to our inability to differentiate graphs visually appropriate evaluation is of crucial importance. Future research directions include benchmarking of graph kernels in combination with MMD and the development of procedures that can automatically derive the appropriate hyperparameters in a data-driven fashion.

A further similar direction of research is to reduce the reliance on graph descriptors. Using graph descriptor functions is similar to engineering a set of features for differentiating graphs. This selection necessitates some assumptions on the graph distribution and for every graph descriptor function, we can construct examples where the similarity of graph descriptors would mislead us into assuming graphs would be similar. The problem is thus highly related to determining graph isomorphism and graph neural network expressivity. If we had a graph descriptor that is the same only if two graphs are the same, we would have a perfect isomorphism test. As the graph isomorphism problem is of complexity NP, it is impossible that such a function would exist and run in polynomial time (assuming that $P \neq NP$). Thus, it will be necessary to rely on heuristics to evaluate graph similarity. One such heuristic could be training models on a proxy task of predicting the graph edit distance between two graphs [180]. If a model is trained on a sufficiently diverse set of graphs, one could hope that it transfers to new distributions and could be used to evaluate generated graphs.

* * *

This thesis was motivated by the potential of bridging the gap between machine learning methods and applications in the life sciences and clinic. The goal was to show how representation learning models can be adapted and extended to better tackle problems and data types present in these domains. Throughout the thesis, it became clear that while representation learning can be a powerful tool for medicine and biology, it might miss the mark in the real world if not appropriately evaluated at the development stage. In order to develop models that truly have an impact, machine learning researchers must tightly work together with practitioners and combine knowledge from both worlds. I believe that this is the elemental challenge that, when addressed, can improve the state of care, accelerate science, and ultimately improve human lives.

Acronyms

CDF	Cummulative Distribution Function
CNN	Convolutional Neural Network
DNN	Deep Neural Network
GCS	Glasgow Coma Score
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
MEWS	Modified Early Warning Score
MLP	Multi Layer Perceptron
MPNN	Message Passing Neural Network
NEWS	National Early Warning Score
PCA	Principal Component Analysis
PDE	Partial Differential Equation
qSOFA	Quick Sequential Organ Failure Assessment
RNN	Recurrent Neural Network
SBM	Stochastic Block Model
SGD	Stochastic Gradient Descent
SIRS	Systemic Inflammatory Response Syndrome
SOFA	Sequential Organ Failure Assessment
TDA	Topological Data Analysis
TOGL	Topological Graph Layer
TopoAE	Topological AutoEncoders
WL	Weisfeiler–Leman

Bibliography

1. C. Aggarwal, G. He, and P. Zhao. "Edge classification in networks". In: *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE. 2016, pp. 1038–1049.
2. B. Alberts. *Molecular biology of the cell*. WW Norton & Company, 2017.
3. B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey. "Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning". *Nature biotechnology* 33:8, 2015, pp. 831–838.
4. M. Allgöwer and C. Burri. "Schockindex". *DMW-Deutsche Medizinische Wochenschrift* 92:43, 1967, pp. 1947–1950.
5. M. Amodio, D. Van Dijk, K. Srinivasan, W. S. Chen, H. Mohsen, K. R. Moon, A. Campbell, Y. Zhao, X. Wang, M. Venkataswamy, et al. "Exploring single-cell data with deep multitasking neural networks". *Nature methods* 16:11, 2019, pp. 1139–1145.
6. N. Aronszajn. "Theory of reproducing kernels". *Transactions of the American mathematical society* 68:3, 1950, pp. 337–404.
7. V. Arvind, F. Fuhlbrück, J. Köbler, and O. Verbitsky. "On weisfeiler-leman invariance: subgraph counts and related graph properties". *Journal of Computer and System Sciences* 113, 2020, pp. 42–59.
8. A. Avati, M. Seneviratne, E. Xue, Z. Xu, B. Lakshminarayanan, and A. M. Dai. "BEDS-Bench: Behavior of EHR-models under Distributional Shift–A Benchmark". *arXiv preprint arXiv:2107.08189*, 2021.
9. J. L. Ba, J. R. Kiros, and G. E. Hinton. "Layer normalization". 2016. arXiv: `1607.06450 [cs.LG]`.
10. L. Babai and L. Kucera. "Canonical labelling of graphs in linear average time". In: *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*. IEEE. 1979, pp. 39–46.

11. T. Bachlechner, B. P. Majumder, H. Mao, G. Cottrell, and J. McAuley. "ReZero is all you need: fast convergence at large depth". In: *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*. Ed. by C. de Campos and M. H. Maathuis. Vol. 161. Proceedings of Machine Learning Research. PMLR, 2021, pp. 1352–1361.
12. M. T. Bahadori and Z. Lipton. "Temporal-clustering invariance in irregular healthcare time series". In: *ACM CHIL 2020*. 2020.
13. S. Bai, J. Z. Kolter, and V. Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling". Preprint. 2018. arXiv: 1803.01271 [cs.LG].
14. M. Barish, S. Bolourani, L. F. Lau, S. Shah, and T. P. Zanos. "External validation demonstrates limited clinical utility of the interpretable mortality prediction model for patients with COVID-19". *Nature Machine Intelligence* 3:1, 2021, pp. 25–27.
15. P. Battaglia, J. B. C. Hamrick, V. Bapst, A. Sanchez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. Allen, C. Nash, V. J. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. "Relational inductive biases, deep learning, and graph networks". Preprint. 2018. arXiv: 1806.01261 [cs.LG].
16. M. Belkin and P. Niyogi. "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering". In: *Advances in Neural Information Processing Systems 14*. 2002, pp. 585–591.
17. P. Bendich, P. Bubenik, and A. Wagner. "Stabilizing the unstable output of persistent homology computations". *Journal of Applied and Computational Topology* 4:2, 2020, pp. 309–338.
18. Y. Bengio, A. Courville, and P. Vincent. "Representation learning: A review and new perspectives". *IEEE transactions on pattern analysis and machine intelligence* 35:8, 2013, pp. 1798–1828.
19. N. Bennett, D. Plečko, I.-F. Ukor, N. Meinshausen, and P. Bühlmann. "ricu: R's Interface to Intensive Care Data". Preprint. 2021. arXiv: 2108.00796 [cs.LG].
20. C. M. Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.
21. M. Blondel, O. Teboul, Q. Berthet, and J. Djolonga. "Fast differentiable sorting and ranking". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 950–959.
22. R. C. Bone, R. A. Balk, F. B. Cerra, R. P. Dellinger, A. M. Fein, W. A. Knaus, R. M. Schein, and W. J. Sibbald. "Definitions for sepsis and organ failure and guidelines for the use of innovative therapies in sepsis". *Chest* 101:6, 1992, pp. 1644–1655.

23. E. V. Bonilla, K. M. Chai, and C. Williams. "Multi-task Gaussian process prediction". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2008, pp. 153–160.

24. K. Borgwardt, E. Ghisu, F. Llinares-López, L. O'Bray, B. Rieck, et al. "Graph kernels: State-of-the-art and future challenges". *Foundations and Trends® in Machine Learning* 13:5-6, 2020, pp. 531–712.

25. K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola. "Integrating structured biological data by kernel maximum mean discrepancy". *Bioinformatics* 22:14, 2006, e49–e57.

26. K. M. Borgwardt and H.-P. Kriegel. "Shortest-path kernels on graphs". In: *Fifth IEEE international conference on data mining (ICDM'05)*. IEEE. 2005, 8–pp.

27. G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein. "Improving graph neural network expressivity via subgraph isomorphism counting". Preprint. 2020. arXiv: 2006.09252 [cs.LG].

28. C. Bouza, T. Lopez-Cuadrado, and J. Amate-Blanco. "Use of explicit ICD9-CM codes to identify adult severe sepsis: impacts on epidemiological estimates". *Critical Care* 20:1, 2016, pp. 1–12.

29. F. A. Bozza, J. I. Salluh, A. M. Japiassu, M. Soares, E. F. Assis, R. N. Gomes, M. T. Bozza, H. C. Castro-Faria-Neto, and P. T. Bozza. "Cytokine profiles as markers of disease severity in sepsis: a multiplex analysis". *Critical Care* 11:2, 2007, R49.

30. X. Bresson and T. Laurent. "Residual gated graph convnets". Preprint. 2017. arXiv: 1711.07553 [cs.LG].

31. B. C. Brown, A. L. Caterini, B. L. Ross, J. C. Cresswell, and G. Loaiza-Ganem. "The Union of Manifolds Hypothesis and its Implications for Deep Generative Modelling". *arXiv preprint arXiv:2207.02862*, 2022.

32. G. Brunner, Y. Liu, D. Pascual, O. Richter, M. Ciaramita, and R. Wattenhofer. "On Identifiability in Transformers". In: *International Conference on Learning Representations*. 2019.

33. P. Bryant, G. Pozzati, and A. Elofsson. "Improved prediction of protein-protein interactions using AlphaFold2". *Nature communications* 13:1, 2022, pp. 1–11.

34. D. Buffelli and F. Vandin. "The Impact of Global Structural Information in Graph Neural Networks Applications". *Data* 7:1, 2022, p. 10.

35. D. Burago, Y. Burago, and S. Ivanov. *A course in metric geometry*. Vol. 33. Graduate Studies in Mathematics. American Mathematical Society, 2001.

36. H. Burdick, E. Pino, D. Gabel-Comeau, A. McCoy, C. Gu, J. Roberts, S. Le, J. Slote, E. Pellegrini, A. Green-Saxena, et al. "Effect of a sepsis prediction algorithm on patient mortality, length of stay and readmission: a prospective multicentre clinical outcomes evaluation of real-world patient data from US hospitals". *BMJ health & care informatics* 27:1, 2020.

37. M. Campbell, A. J. Hoane Jr, and F.-h. Hsu. "Deep blue". *Artificial intelligence* 134:1-2, 2002, pp. 57–83.

38. Y.-H. Cao and J. Wu. "A Random CNN Sees Objects: One Inductive Bias of CNN and Its Applications". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 1. 2022, pp. 194–202.

39. G. Carlsson and A. Zomorodian. "The theory of multidimensional persistence". *Discrete & Computational Geometry* 42:1, 2009, pp. 71–93.

40. M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. "Emerging properties in self-supervised vision transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9650–9660.

41. M. Carriere, F. Chazal, M. Glisse, Y. Ike, H. Kannan, and Y. Umeda. "Optimizing persistent homology based functions". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 1294–1303.

42. M. Carrière, F. Chazal, Y. Ike, T. Lacombe, M. Royer, and Y. Umeda. "PersLay: A Neural Network Layer for Persistence Diagrams and New Graph Topological Signatures". In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*. Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2786–2796.

43. S. Carter, Z. Armstrong, L. Schubert, I. Johnson, and C. Olah. "Activation Atlas". *Distill*, 2019. https://distill.pub/2019/activation-atlas.

44. T. Chari, J. Banerjee, and L. Pachter. "The specious art of single-cell genomics". *BioRxiv*, 2021.

45. F. Chazal, D. Cohen-Steiner, and Q. Mérigot. "Geometric Inference for Probability Measures". *Foundations of Computational Mathematics* 11:6, 2011, pp. 733–751.

46. F. Chazal, V. de Silva, and S. Y. Oudot. "Persistence stability for geometric complexes". *Geometriæ Dedicata* 173:1, 2014, pp. 193–214.

47. Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu. "Recurrent neural networks for multivariate time series with missing values". *Scientific reports* 8:1, 2018, p. 6085.

48. D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun. “Measuring and relieving the over-smoothing problem for graph neural networks from the topological view”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 3438–3445.

49. Z. Chen, L. Chen, S. Villar, and J. Bruna. “Can Graph Neural Networks Count Substructures?” In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 10383–10395.

50. Z. Chen, S. Villar, L. Chen, and J. Bruna. “On the equivalence between graph isomorphism testing and function approximation with GNNs”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

51. K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. Preprint. 2014. arXiv: `1406.1078 [cs.LG]`.

52. F. R. K. Chung. *Spectral Graph Theory*. Vol. 92. CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

53. D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. “Extending persistence using Poincaré and Lefschetz duality”. *Foundations of Computational Mathematics* 9:1, 2009, pp. 79–103.

54. D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. “Stability of persistence diagrams”. *Discrete & Computational Geometry* 37:1, 2007, pp. 103–120.

55. C. M. Cutillo, K. R. Sharma, L. Foschini, S. Kundu, M. Mackintosh, and K. D. Mandl. “Machine intelligence in healthcare—perspectives on trustworthiness, explainability, usability, and transparency”. *NPJ digital medicine* 3:1, 2020, pp. 1–5.

56. M. Cuturi, O. Teboul, and J.-P. Vert. “Differentiable Ranking and Sorting using Optimal Transport”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

57. E. R. van Dam and W. H. Haemers. “Which graphs are determined by their spectrum?” *Linear Algebra and its Applications* 373, 2003, pp. 241–272.

58. K. De Bruyne, B. Slabbinck, W. Waegeman, P. Vauterin, B. De Baets, and P. Vandamme. “Bacterial Species Identification from MALDI-TOF Mass Spectra through Data Analysis and Machine Learning”. en. *Systematic and Applied Microbiology* 34:1, 2011, pp. 20–29. ISSN: 07232020.

59. K. De Bruyne, B. Slabbinck, W. Waegeman, P. Vauterin, B. De Baets, and P. Vandamme. "Bacterial species identification from MALDI-TOF mass spectra through data analysis and machine learning". *Systematic and applied microbiology* 34:1, 2011, pp. 20–29.

60. R. P. Dellinger, M. M. Levy, A. Rhodes, D. Annane, H. Gerlach, S. M. Opal, J. E. Sevransky, C. L. Sprung, I. S. Douglas, R. Jaeschke, T. M. Osborn, M. E. Nunnally, S. R. Townsend, K. Reinhart, R. M. Kleinpell, D. C. Angus, C. S. Deutschman, F. R. Machado, G. D. Rubenfeld, S. A. Webb, R. J. Beale, J.-L. Vincent, and R. Moreno. "Surviving Sepsis Campaign: International Guidelines for Management of Severe Sepsis and Septic Shock 2012". *Critical Care Medicine* 41:2, 2013, pp. 580–637.

61. K. A. Dill, S. B. Ozkan, M. S. Shell, and T. R. Weikl. "The protein folding problem". *Annual review of biophysics* 37, 2008, p. 289.

62. J. Ding, A. Condon, and S. P. Shah. "Interpretable dimensionality reduction of single cell transcriptome data with deep generative models". *Nature communications* 9:1, 2018, pp. 1–13.

63. S. Dodge and L. Karam. "A study and comparison of human and deep learning recognition performance under visual distortions". In: *2017 26th international conference on computer communication and networks (ICCCN)*. IEEE. 2017, pp. 1–7.

64. R. M. Dudley. *Real analysis and probability*. CRC Press, 2018.

65. G. Duke, J. Briedis, and J. Green. "Survival of critically ill medical patients is time-critical". *Critical Care and Resuscitation* 6:4, 2004.

66. V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson. "Benchmarking graph neural networks". Preprint. 2020. arXiv: `2003.00982 [cs.LG]`.

67. G. K. Dziugaite, D. M. Roy, and Z. Ghahramani. "Training generative neural networks via maximum mean discrepancy optimization". In: *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*. 2015, pp. 258–267.

68. H. Edelsbrunner, J. Harer, et al. "Persistent homology-a survey". *Contemporary mathematics* 453, 2008, pp. 257–282.

69. H. Edelsbrunner, D. Letscher, and A. Zomorodian. "Topological persistence and simplification". In: *Proceedings 41st annual symposium on foundations of computer science*. IEEE. 2000, pp. 454–463.

70. J. M. Ehrenfeld and M. Cannesson. *Monitoring technologies in acute care environments*. Springer, 2014.

71. A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. "Dermatologist-level classification of skin cancer with deep neural networks". *nature* 542:7639, 2017, pp. 115–118.

72. A.-C. Falk and E.-M. Wallin. "Quality of patient care in the critical care unit in relation to nurse patient ratio: A descriptive study". *Intensive and Critical Care Nursing* 35, 2016, pp. 74–79.

73. A. Farahani, S. Voghoei, K. Rasheed, and H. R. Arabnia. "A brief review of domain adaptation". *Advances in Data Science and Information Engineering*, 2021, pp. 877–894.

74. R. Ferrer, I. Martin-Loeches, G. Phillips, T. M. Osborn, S. Townsend, R. P. Dellinger, A. Artigas, C. Schorr, and M. M. Levy. "Empiric antibiotic treatment reduces mortality in severe sepsis and septic shock from the first hour: results from a guideline-based performance improvement program". *Critical Care Medicine* 42:8, 2014, pp. 1749–1755.

75. S. D. Fihn, J. Francis, C. Clancy, C. Nielson, K. Nelson, J. Rumsfeld, T. Cullen, J. Bates, and G. L. Graham. "Insights from advanced analytics at the Veterans Health Administration". *Health affairs* 33:7, 2014, pp. 1203–1211.

76. M. A. Finzi, R. Bondesan, and M. Welling. "Probabilistic Numeric Convolutional Neural Networks". In: *International Conference on Learning Representations*. 2021.

77. C. Fleischmann-Struzek, D. O. Thomas-Rüddel, A. Schettler, D. Schwarzkopf, A. Stacke, C. W. Seymour, C. Haas, U. Dennler, and K. Reinhart. "Comparing the validity of different ICD coding abstraction strategies for sepsis case identification in German claims data". *PLoS One* 13:7, 2018, e0198847.

78. L. M. Fleuren, T. L. Klausch, C. L. Zwager, L. J. Schoonmade, T. Guo, L. F. Roggeveen, E. L. Swart, A. R. Girbes, P. Thoral, A. Ercole, et al. "Machine learning for the prediction of sepsis: a systematic review and meta-analysis of diagnostic test accuracy". *Intensive Care Medicine*, 2020, pp. 1–18.

79. K. Fukumizu, A. Gretton, G. Lanckriet, B. Schölkopf, and B. K. Sriperumbudur. "Kernel Choice and Classifiability for RKHS Embeddings of Probability Distributions". In: *Advances in Neural Information Processing Systems*. Ed. by Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta. Vol. 22. Curran Associates, Inc., 2009.

80. J. Futoma, S. Hariharan, and K. Heller. "Learning to Detect Sepsis with a Multitask Gaussian Process RNN Classifier". In: *International Conference on Machine Learning (ICML)*. 2017, pp. 1174–1182.

81. M. Gameiro, Y. Hiraoka, and I. Obayashi. "Continuation of point clouds via persistence diagrams". *Physica D: Nonlinear Phenomena* 334, 2016, pp. 118–132.

82. S. Gando, A. Shiraishi, T. Abe, S. Kushimoto, T. Mayumi, S. Fujishima, A. Hagiwara, Y. Shiino, S.-i. Shiraishi, T. Hifumi, et al. "The SIRS criteria have better performance for predicting infection than qSOFA scores in the emergency department". *Scientific reports* 10:1, 2020, pp. 1–9.

83. M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. A. Eslami. "Conditional Neural Processes". In: *International Conference on Machine Learning*. 2018, pp. 1690–1699.

84. S. Gibb and K. Strimmer. "MALDIquant: a versatile R package for the analysis of mass spectrometry data". *Bioinformatics* 28:17, 2012, pp. 2270–2271.

85. S. Gibb and K. Strimmer. "MALDIquant: a versatile R package for the analysis of mass spectrometry data". *Bioinformatics* 28:17, 2012, pp. 2270–2271.

86. A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. "PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals". *Circulation* 101:23, 2000, e215–e220.

87. I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

88. A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. "A kernel method for the two-sample-problem". *Advances in neural information processing systems* 19, 2006, pp. 513–520.

89. H. Guan and M. Liu. "Domain adaptation for medical image analysis: a survey". Preprint. 2021. arXiv: 2102.09508 [cs.LG].

90. X. Guo and L. Zhao. "A systematic survey on deep generative models for graph generation". Preprint. 2020. arXiv: 2007.06686 [cs.LG].

91. H. Harutyunyan, H. Khachatrian, D. C. Kale, G. Ver Steeg, and A. Galstyan. "Multitask learning and benchmarking with clinical time series data". *Scientific Data* 6:1, 2019, p. 96. ISSN: 2052-4463.

92. K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

93. F. Hensel, M. Moor, and B. Rieck. "A Survey of Topological Machine Learning Methods". *Frontiers in Artificial Intelligence* 4, 2021.

94. J. Hester, T. S. Youn, E. Trifilio, C. P. Robinson, M.-A. Babi, P. Ameli, W. Roth, S. Gatica, M. A. Pizzi, A. Gennaro, et al. "The Modified Early Warning Score: A Useful Marker of Neurological Worsening but Unreliable Predictor of Sepsis in the Neurocritically Ill-A Retrospective Cohort Study". *Critical care explorations* 3:5, 2021, e0386.

95. M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

96. S. Hochreiter and J. Schmidhuber. "Long short-term memory". *Neural computation* 9:8, 1997, pp. 1735–1780.

97. C. Hofer, F. Graf, B. Rieck, M. Niethammer, and R. Kwitt. "Graph filtration learning". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 4314–4323.

98. C. Hofer, R. Kwitt, M. Niethammer, and M. Dixit. "Connectivity-Optimized Representation Learning via Persistent Homology". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 2751–2760.

99. C. Hofer, R. Kwitt, M. Niethammer, and A. Uhl. "Deep Learning with Topological Signatures". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017, pp. 1634–1644.

100. C. D. Hofer, R. Kwitt, and M. Niethammer. "Learning Representations of Persistence Barcodes." *J. Mach. Learn. Res.* 20:126, 2019, pp. 1–45.

101. M. Horn, M. Moor, C. Bock, B. Rieck, and K. Borgwardt. "Set functions for time series". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 4353–4363.

102. M. Horn*, E. D. Brouwer*, M. Moor, Y. Moreau, B. Rieck, and K. Borgwardt. "Topological Graph Neural Networks". In: *International Conference on Learning Representations*. 2022.

103. M. Horn*, K. Shridhar*, E. Groenewald*, and P. F. Baumann*. "Translational Equivariance in Kernelizable Attention". Preprint. 2021. arXiv: `2102.07680 [cs.LG]`.

104. D. J. Horton, K. K. Graves, P. V. Kukhareva, S. A. Johnson, M. Cedillo, M. Sanford, W. A. Dunson Jr, M. White, D. Roach, J. J. Arego, et al. "Modified early warning score-based clinical decision support: cost impact and clinical outcomes in sepsis". *JAMIA open* 3:2, 2020, pp. 261–268.

105. R. S. Hotchkiss, L. L. Moldawer, S. M. Opal, K. Reinhart, I. R. Turnbull, and J. L. Vincent. “Sepsis and septic shock”. *Nature Reviews Disease Primers* 2, 2016.

106. H. Hotelling. “Analysis of a complex of statistical variables into principal components.” *Journal of educational psychology* 24:6, 1933, p. 417.

107. S.-M. Hsieh, C.-C. Hsu, and L.-F. Hsu. “Efficient method to perform isomorphism testing of labeled graphs”. In: *International Conference on Computational Science and Its Applications*. Springer. 2006, pp. 422–431.

108. S. L. Hyland, M. Faltys, M. Hüser, X. Lyu, T. Gumbsch, C. Esteban, C. Bock, M. Horn, M. Moor, B. Rieck, et al. “Early prediction of circulatory failure in the intensive care unit using machine learning”. *Nature medicine* 26:3, 2020, pp. 364–373.

109. S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

110. S. Jain and B. C. Wallace. “Attention is not Explanation”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 3543–3556.

111. B. C. James and M. E. H. Hammond. “The challenge of variation in medical practice”. *Archives of pathology & laboratory medicine* 124:7, 2000, pp. 1001–1003.

112. Y. Ji, M. Lotfollahi, F. A. Wolf, and F. J. Theis. “Machine learning for perturbational single-cell omics”. *Cell Systems* 12:6, 2021, pp. 522–537.

113. A. E. W. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark. “MIMIC-III, a freely accessible critical care database”. *Scientific Data* 3, 2016.

114. W. B. Johnson and J. Lindenstrauss. “Extensions of Lipschitz mappings into a Hilbert space 26”. *Contemporary mathematics* 26, 1984.

115. M. Jones. “NEWSDIG: The national early warning score development and implementation group”. *Clinical Medicine* 12:6, 2012, p. 501.

116. J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. “Highly accurate protein structure prediction with AlphaFold”. *Nature* 596:7873, 2021, pp. 583–589.

117. A. Karpathy. “Stanford university cs231n: Convolutional neural networks for visual recognition”. *URL: http://cs231n. stanford. edu/syllabus. html*, 2018, pp. 13–35.

118. K. M. Kaukonen, M. Bailey, S. Suzuki, D. Pilcher, and R. Bellomo. "Mortality related to severe sepsis and septic shock among critically ill patients in Australia and New Zealand, 2000-2012." *JAMA* 311:13, 2014, pp. 1308–1316.

119. G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. "Lightgbm: A highly efficient gradient boosting decision tree". *Advances in neural information processing systems* 30, 2017, pp. 3146–3154.

120. O. Keskin, N. Tuncbag, and A. Gursoy. "Predicting protein–protein interactions from the molecular to the proteome level". *Chemical reviews* 116:8, 2016, pp. 4884–4909.

121. D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. 2014.

122. T. N. Kipf and M. Welling. "Semi-supervised classification with graph convolutional networks". Preprint. 2016. arXiv: 1609.02907 [cs.LG].

123. H. Kitano. "Computational systems biology". *Nature* 420:6912, 2002, pp. 206–210.

124. M. Komorowski, L. A. Celi, O. Badawi, A. C. Gordon, and A. A. Faisal. "The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care". *Nature medicine* 24:11, 2018, pp. 1716–1720.

125. N. M. Kriege, P.-L. Giscard, and R. Wilson. "On valid optimal assignment kernels and applications to graph classification". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016, pp. 1623–1631.

126. A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". *Advances in neural information processing systems* 25, 2012, pp. 1097–1105.

127. E. Laparra, A. Mascio, S. Velupillai, and T. Miller. "A Review of Recent Work in Transfer Learning and Domain Adaptation for Natural Language Processing of Electronic Health Records". *Yearbook of Medical Informatics* 30:01, 2021, pp. 239–244.

128. F. F. Larsen and J. A. Petersen. "Novel biomarkers for sepsis: A narrative review". *European Journal of Internal Medicine* 45, 2017, pp. 46–50.

129. Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". *nature* 521:7553, 2015, pp. 436–444.

130. Y. LeCun, C. Cortes, and C. Burges. "MNIST handwritten digit database". *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2, 2010.

131. A. B. Lee, K. S. Pedersen, and D. Mumford. "The nonlinear statistics of high-contrast patches in natural images". *International Journal of Computer Vision* 54:1, 2003, pp. 83–103.

132. J. A. Lee and M. Verleysen. "Quality assessment of dimensionality reduction: Rank-based criteria". *Neurocomputing* 72:7, 2009, pp. 1431–1443.

133. J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh. "Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Long Beach, California, USA, 2019, pp. 3744–3753.

134. M. Leone, P. Asfar, P. Radermacher, J.-L. Vincent, and C. Martin. "Optimizing mean arterial pressure in septic shock: a critical reappraisal of the literature". *Critical Care* 19:1, 2015, pp. 1–7.

135. M. M. Levy, M. P. Fink, J. C. Marshall, E. Abraham, D. Angus, D. Cook, J. Cohen, S. M. Opal, J.-L. Vincent, and G. Ramsay. "2001 sccm/esicm/accp/ats/sis international sepsis definitions conference". *Intensive care medicine* 29:4, 2003, pp. 530–538.

136. S. C.-X. Li and B. M. Marlin. "Classification of Sparse and Irregularly Sampled Time Series with Mixtures of Expected Gaussian Kernels and Random Features." In: *UAI*. 2015, pp. 484–493.

137. S. C.-X. Li and B. M. Marlin. "A scalable end-to-end Gaussian process adapter for irregularly sampled time series classification". In: *Advances In Neural Information Processing Systems 29*. 2016, pp. 1804–1812.

138. Y. Li, K. Swersky, and R. Zemel. "Generative moment matching networks". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1718–1727.

139. R. Liao, Y. Li, Y. Song, S. Wang, W. Hamilton, D. K. Duvenaud, R. Urtasun, and R. Zemel. "Efficient Graph Generation with Graph Recurrent Attention Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

140. T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. "Feature pyramid networks for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

141. Z. C. Lipton, D. Kale, and R. Wetzel. "Directly modeling missing data in sequences with rnns: Improved classification of clinical time series". In: *Machine Learning for Healthcare Conference*. 2016, pp. 253–270.

142. J. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax Weiss, and B. Lakshminarayanan. "Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 7498–7512.

143. Q. Liu, M. Allamanis, M. Brockschmidt, and A. Gaunt. "Constrained graph variational autoencoders for molecule design". *Advances in neural information processing systems* 31, 2018.

144. H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. Watkins. "Text Classification using String Kernels". In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press, 2001.

145. L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis. "Dying relu and initialization: Theory and numerical examples". Preprint. 2019. arXiv: 1903.06733 [cs.LG].

146. Z. Lu, E. Ie, and F. Sha. "Mean-Field Approximation to Gaussian-Softmax Integral with Application to Uncertainty Estimation". 2020. arXiv: 2006.07584 [cs.LG].

147. S. M. Lundberg and S.-I. Lee. "A unified approach to interpreting model predictions". In: *Proceedings of the 31st international conference on neural information processing systems*. 2017, pp. 4768–4777.

148. L. J. van der Maaten and G. Hinton. "Visualizing data using t-SNE". *Journal of Machine Learning Research* 9, 2008, pp. 2579–2605.

149. D. J. MacKay and D. J. Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

150. *MALDI Biotyper System, Bruker Daltonics*. https://www.bruker.com/products/mass-spectrometry-and-separations/fda-cleared-maldi-biotyper-usa/overview.html. 2019.

151. H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman. "Provably Powerful Graph Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

152. C. A. Mather, B. J. Werth, S. Sivagnanam, D. J. Sengupta, and S. M. Butler-Wu. "Rapid Detection of Vancomycin-Intermediate *Staphylococcus aureus* by Matrix-Assisted Laser Desorption Ionization–Time of Flight Mass Spectrometry". *Journal of Clinical Microbiology* 54:4, 2016, pp. 883–890.

153. L. McInnes, J. Healy, and J. Melville. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction”. Preprint. 2018. arXiv: 1802.03426 [stat.ML].

154. R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley. “Deep learning for healthcare: review, opportunities and challenges”. *Briefings in bioinformatics* 19:6, 2018, pp. 1236–1246.

155. C. Molnar. *Interpretable machine learning*. Lulu. com, 2020.

156. K. R. Moon, D. van Dijk, Z. Wang, S. Gigante, D. B. Burkhardt, W. S. Chen, K. Yim, A. v. d. Elzen, M. J. Hirn, R. R. Coifman, et al. “Visualizing structure and transitions in high-dimensional biological data”. *Nature biotechnology* 37:12, 2019, pp. 1482–1492.

157. M. Moor, M. Horn, C. Bock, K. Borgwardt, and B. Rieck. “Path Imputation Strategies for Signature Models”. In: *ICML Workshop on the Art of Learning with Missing Values (Artemiss)*. 2020.

158. M. Moor, M. Horn, K. Borgwardt, and B. Rieck. “Challenging euclidean topological autoencoders”. In: *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*. 2020.

159. M. Moor, M. Horn, B. Rieck, D. Roqueiro, and K. Borgwardt. “Early recognition of sepsis with Gaussian process temporal convolutional networks and dynamic time warping”. In: *Machine Learning for Healthcare Conference*. PMLR. 2019, pp. 2–26.

160. M. Moor, B. Rieck, M. Horn, C. R. Jutzeler, and K. Borgwardt. “Early prediction of sepsis in the ICU using machine learning: a systematic review”. *Frontiers in medicine* 8, 2021, p. 607952.

161. M. Moor, B. Rieck, M. Horn, C. R. Jutzeler, and K. Borgwardt. “Early Prediction of Sepsis in the ICU using Machine Learning: A Systematic Review”. *Frontiers in Medicine* 8, 2021.

162. M. Moor*, N. Bennet*, D. Plecko*, M. Horn*, B. Rieck, N. Meinshausen, P. Bühlmann, and K. Borgwardt. “Predicting sepsis in multi-site, multi-national intensive care cohorts using deep learning”. Preprint. 2021. arXiv: 2107.05230 [cs.LG].

163. M. Moor*, M. Horn*, B. Rieck, and K. Borgwardt. “Topological Autoencoders”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7045–7054.

164. C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann. “Tudataset: A collection of benchmark datasets for learning with graphs”. Preprint. 2020. arXiv: 2007.08663 [cs.LG].

165. C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. “Weisfeiler and leman go neural: Higher-order graph neural networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 4602–4609.

166. A. Nazabal, P. M. Olmos, Z. Ghahramani, and I. Valera. “Handling incomplete heterogeneous data using vaes”. *Pattern Recognition* 107, 2020, p. 107501.

167. D. Neil, M. Pfeiffer, and S.-C. Liu. “Phased lstm: Accelerating recurrent network training for long or event-based sequences”. In: *Advances in Neural Information Processing Systems 29*. 2016, pp. 3882–3890.

168. J. A. Nelder and R. W. Wedderburn. “Generalized linear models”. *Journal of the Royal Statistical Society: Series A (General)* 135:3, 1972, pp. 370–384.

169. T. Q. Nguyen and J. Salazar. “Transformers without Tears: Improving the Normalization of Self-Attention”. In: *Proceedings of the 16th International Conference on Spoken Language Translation*. Association for Computational Linguistics, Hong Kong, 2019.

170. S. J. Nightingale, K. A. Wade, and D. G. Watson. “Can people identify original and manipulated photos of real-world scenes?” *Cognitive research: principles and implications* 2:1, 2017, pp. 1–21.

171. C. Niu, Y. Song, J. Song, S. Zhao, A. Grover, and S. Ermon. “Permutation Invariant Graph Generation via Score-Based Generative Modeling”. In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by S. Chiappa and R. Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 4474–4484.

172. Y. Niv. “Learning task-state representations”. *Nature neuroscience* 22:10, 2019, pp. 1544–1553.

173. Y. Numata, M. Schulzer, R. Van Der Wal, J. Globerman, P. Semeniuk, E. Balka, and J. M. FitzGerald. “Nurse staffing levels and hospital mortality in critical care settings: literature review and meta-analysis”. *Journal of advanced nursing* 55:4, 2006, pp. 435–448.

174. L. O’Bray*, M. Horn*, B. Rieck, and K. Borgwardt. “Evaluation Metrics for Graph Generative Models: Problems, Pitfalls, and Practical Solutions”. In: *International Conference on Learning Representations*. 2022.

175. OECD. *Geographic Variations in Health Care*. 2014, p. 416.

176. K. Oono and T. Suzuki. “Graph Neural Networks Exponentially Lose Expressive Power for Node Classification”. In: *International Conference on Learning Representations*. 2020.

177. A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. “Wavenet: A generative model for raw audio”. Preprint. 2016. arXiv: 1609.03499 [cs.LG].

178. R. M. Ortiz, R. E. Cilley, and R. H. Bartlett. “Extracorporeal membrane oxygenation in pediatric respiratory failure”. *Pediatric Clinics of North America* 34:1, 1987, pp. 39–46.

179. M. Osthoff, N. Gürtler, S. Bassetti, G. Balestra, S. Marsch, H. Pargger, M. Weisser, and A. Egli. “Impact of MALDI-TOF-MS-based identification directly from positive blood cultures on patient management: a controlled clinical trial”. *Clinical Microbiology and Infection* 23:2, 2017, pp. 78–85.

180. B. Paassen, D. Grattarola, D. Zambon, C. Alippi, and B. E. Hammer. “Graph Edit Networks”. In: *International Conference on Learning Representations*. 2021.

181. R. Pascanu, T. Mikolov, and Y. Bengio. “On the difficulty of training recurrent neural networks”. In: *International conference on machine learning*. PMLR. 2013, pp. 1310–1318.

182. S. Peake, R. Bellomo, P. Cameron, A. Cross, A. Delaney, S. Finfer, C. George, A. Higgins, D. Jones, J. Moran, J. Myburgh, G. Syres, S. Webb, and P. Williams. “The outcome of patients with sepsis and septic shock presenting to emergency departments in Australia and New Zealand”. *Critical Care and Resuscitation* 9:1, 2007, pp. 8–18.

183. K. Pearson. “LIII. On lines and planes of closest fit to systems of points in space”. *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2:11, 1901, pp. 559–572.

184. G. Peyré. “Manifold models for signals and images”. *Computer vision and image understanding* 113:2, 2009, pp. 249–260.

185. P. J. Phillips, A. N. Yates, Y. Hu, C. A. Hahn, E. Noyes, K. Jackson, J. G. Cavazos, G. Jeckeln, R. Ranjan, S. Sankaranarayanan, et al. “Face recognition accuracy of forensic examiners, superrecognizers, and face recognition algorithms”. *Proceedings of the National Academy of Sciences* 115:24, 2018, pp. 6171–6176.

186. T. J. Pollard, A. E. Johnson, J. D. Raffa, L. A. Celi, R. G. Mark, and O. Badawi. “The eICU Collaborative Research Database, a freely available multi-center database for critical care research”. *Scientific data* 5:1, 2018, pp. 1–13.

187. A. Poulenard, P. Skraba, and M. Ovsjanikov. “Topological function optimization for continuous shape matching”. In: *Computer Graphics Forum*. Vol. 37. 5. Wiley Online Library. 2018, pp. 13–25.

188. L. Pruinelli, B. L. Westra, P. Yadav, A. Hoff, M. Steinbach, V. Kumar, C. W. Delaney, and G. Simon. “Delay within the 3-hour surviving sepsis campaign guideline on mortality for patients with severe sepsis and septic shock”. *Critical Care Medicine* 46:4, 2018, p. 500.

189. Z. Qian, W. R. Zame, L. M. Fleuren, P. Elbers, and M. van der Schaar. "Integrating expert ODEs into Neural ODEs: Pharmacology and disease progression". In: *Thirty-Fifth Conference on Neural Information Processing Systems*. 2021.

190. A. Rajkomar, E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun, et al. "Scalable and accurate deep learning with electronic health records". *NPJ digital medicine* 1:1, 2018, pp. 1–10.

191. C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.

192. J. Reininghaus, S. Huber, U. Bauer, and R. Kwitt. "A Stable Multi-scale Kernel for Topological Machine Learning". In: *CVPR*. 2015, pp. 4741–4748.

193. J. A. Reuter, D. V. Spacek, and M. P. Snyder. "High-throughput sequencing technologies". *Molecular cell* 58:4, 2015, pp. 586–597.

194. M. A. Reyna, C. S. Josef, R. Jeter, S. P. Shashikumar, M. B. Westover, S. Nemati, G. D. Clifford, and A. Sharma. "Early prediction of sepsis from clinical data: the PhysioNet/Computing in Cardiology Challenge 2019". *Critical care medicine* 48:2, 2020, pp. 210–217.

195. B. Rieck. "Persistent Homology in Multivariate Data Visualization". PhD thesis. Ruprecht-Karls-Universität Heidelberg, 2017.

196. B. Rieck and H. Leitte. "Persistent homology for the evaluation of dimensionality reduction schemes". In: *Computer Graphics Forum*. Vol. 34. 3. Wiley Online Library. 2015, pp. 431–440.

197. B. Rieck, M. Togninalli, C. Bock, M. Moor, M. Horn, T. Gumbsch, and K. Borgwardt. "Neural Persistence: A Complexity Measure for Deep Neural Networks Using Algebraic Topology". In: *International Conference on Learning Representations*. 2019.

198. N. Rieke, J. Hancox, W. Li, F. Milletari, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, et al. "The future of digital health with federated learning". *NPJ digital medicine* 3:1, 2020, pp. 1–7.

199. J. Roe. *Elliptic operators, topology and asymptotic methods*. Second. Chapman & Hall/CRC, 1988.

200. Y. Rubanova, R. T. Q. Chen, and D. K. Duvenaud. "Latent Ordinary Differential Equations for Irregularly-Sampled Time Series". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

201. C. Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. *Nature Machine Intelligence* 1:5, 2019, pp. 206–215.

202. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. “ImageNet Large Scale Visual Recognition Challenge”. *International Journal of Computer Vision (IJCV)* 115:3, 2015, pp. 211–252.

203. Y. Sakr, U. Jaschinski, X. Wittebole, T. Szakmany, J. Lipman, S. A. Ñamendys-Silva, I. Martin-Loeches, M. Leone, M.-N. Lupu, J.-L. Vincent, et al. “Sepsis in intensive care unit patients: worldwide data from the intensive care over nations audit”. In: *Open forum infectious diseases*. Vol. 5. 12. Oxford University Press US. 2018, ofy313.

204. T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. “Evolution strategies as a scalable alternative to reinforcement learning”. Preprint. 2017. arXiv: 1703.03864 [cs.LG].

205. F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. “The graph neural network model”. *IEEE transactions on neural networks* 20:1, 2008, pp. 61–80.

206. A. M. Schäfer and H. G. Zimmermann. “Recurrent neural networks are universal approximators”. In: *International Conference on Artificial Neural Networks*. Springer. 2006, pp. 632–640.

207. S. Schamoni, H. A. Lindner, V. Schneider-Lindner, M. Thiel, and S. Riezler. “Leveraging implicit expert knowledge for non-circular machine learning in sepsis prediction”. *Artificial intelligence in medicine* 100, 2019, p. 101725.

208. B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio. “Toward causal representation learning”. *Proceedings of the IEEE* 109:5, 2021, pp. 612–634.

209. B. Schölkopf, A. Smola, and K.-R. Müller. “Nonlinear component analysis as a kernel eigenvalue problem”. *Neural computation* 10:5, 1998, pp. 1299–1319.

210. B. Schölkopf, A. J. Smola, F. Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

211. B. Schölkopf, K. Tsuda, and J.-P. Vert. *Kernel methods in computational biology*. MIT press, 2004.

212. P. Schulam and S. Saria. “Reliable decision support using counterfactual models”. *Advances in neural information processing systems* 30, 2017.

213. S. Serrano and N. A. Smith. "Is Attention Interpretable?" In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 2931–2951.

214. C. W. Seymour, F. Gesten, H. C. Prescott, M. E. Friedrich, T. J. Iwashyna, G. S. Phillips, S. Lemeshow, T. Osborn, K. M. Terry, and M. M. Levy. "Time to treatment and mortality during mandated emergency care for sepsis". *New England Journal of Medicine* 376:23, 2017, pp. 2235–2244.

215. C. W. Seymour, V. X. Liu, T. J. Iwashyna, F. M. Brunkhorst, T. D. Rea, A. Scherag, G. Rubenfeld, J. M. Kahn, M. Shankar-Hari, M. Singer, et al. "Assessment of clinical criteria for sepsis: for the Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3)". *Jama* 315:8, 2016, pp. 762–774.

216. P. Shaw, J. Uszkoreit, and A. Vaswani. "Self-Attention with Relative Position Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 2018, pp. 464–468.

217. N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. "Weisfeiler-Lehman graph kernels." *Journal of Machine Learning Research* 12:9, 2011.

218. D. W. Shimabukuro, C. W. Barton, M. D. Feldman, S. J. Mataraso, and R. Das. "Effect of a machine learning-based severe sepsis prediction algorithm on patient survival and hospital length of stay: a randomised clinical trial". *BMJ open respiratory research* 4:1, 2017, e000234.

219. S. N. Shukla and B. Marlin. "Interpolation-Prediction Networks for Irregularly Sampled Time Series". In: *International Conference on Learning Representations*. 2019.

220. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search". *nature* 529:7587, 2016, pp. 484–489.

221. M. Simonovsky and N. Komodakis. "Graphvae: Towards generation of small graphs using variational autoencoders". In: *International conference on artificial neural networks*. Springer. 2018, pp. 412–422.

222. M. Singer, C. S. Deutschman, C. W. Seymour, M. Shankar-Hari, D. Annane, M. Bauer, R. Bellomo, G. R. Bernard, J.-D. Chiche, C. M. Coopersmith, et al. "The third international consensus definitions for sepsis and septic shock (Sepsis-3)". *Jama* 315:8, 2016, pp. 801–810.

223. S. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. CRC Press, 2018. ISBN: 9780429961113.

224. C. Subbe, M. Kruger, P. Rutherford, and L. Gemmel. “Validation of a modified Early Warning Score in medical admissions”. *QJM* 94:10, 2001, pp. 521–526.

225. D. J. Sutherland, H.-Y. Tung, H. Strathmann, S. De, A. Ramdas, A. Smola, and A. Gretton. “Generative models and model criticism via optimized maximum mean discrepancy”. In: *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. 2017.

226. S. Tariq, S. Lee, H. Kim, Y. Shin, and S. S. Woo. “Detecting both machine and human created fake face images in the wild”. In: *Proceedings of the 2nd international workshop on multimedia privacy and security*. 2018, pp. 81–87.

227. Y. Tay, M. Dehghani, D. Bahri, and D. Metzler. “Efficient transformers: A survey”. Preprint. 2020. arXiv: `2009.06732 [cs.LG]`.

228. J. B. Tenenbaum, V. De Silva, and J. C. Langford. “A global geometric framework for nonlinear dimensionality reduction”. *Science* 290:5500, 2000, pp. 2319–2323.

229. R. Thompson, B. Knyazev, E. Ghalebi, J. Kim, and G. W. Taylor. “On Evaluation Metrics for Graph Generative Models”. In: *International Conference on Learning Representations*. 2022.

230. P. J. Thoral, J. M. Peppink, R. H. Driessen, E. J. Sijbrands, E. J. Kompanje, L. Kaplan, H. Bailey, J. Kesecioglu, M. Cecconi, M. Churpek, et al. “Sharing ICU Patient Data Responsibly Under the Society of Critical Care Medicine/European Society of Intensive Care Medicine Joint Data Science Collaboration: The Amsterdam University Medical Centers Database (AmsterdamUMCdb) Example”. *Critical care medicine* 49:6, 2021, e563.

231. H.-C. Thorsen-Meyer, A. B. Nielsen, A. P. Nielsen, B. S. Kaas-Hansen, P. Toft, J. Schierbeck, T. Strøm, P. J. Chmura, M. Heimann, L. Dybdahl, et al. “Dynamic and explainable machine learning prediction of mortality in patients in the intensive care unit: a retrospective study of high-frequency data in electronic patient records”. *The Lancet Digital Health* 2:4, 2020, e179–e191.

232. M. Togninalli, E. Ghisu, F. Llinares-López, B. Rieck, and K. Borgwardt. “Wasserstein weisfeiler-lehman graph kernels”. *Advances in Neural Information Processing Systems* 32, 2019.

233. E. J. Topol. “High-performance medicine: the convergence of human and artificial intelligence”. *Nature medicine* 25:1, 2019, pp. 44–56.

234. A. Trabelsi, M. Chaabane, and A. Ben-Hur. “Comprehensive evaluation of deep learning architectures for prediction of DNA/RNA sequence binding specificities”. *Bioinformatics* 35:14, 2019, pp. i269–i277.

235. E. L. Tsalik, L. B. Jaggers, S. W. Glickman, R. J. Langley, J. C. van Velkinburgh, L. P. Park, V. G. Fowler, C. B. Cairns, S. F. Kingsmore, and C. W. Woods. "Discriminative value of inflammatory biomarkers for suspected sepsis". *The Journal of Emergency Medicine* 43:1, 2012, pp. 97–106.

236. M. Uhlén, L. Fagerberg, B. M. Hallström, C. Lindskog, P. Oksvold, A. Mardinoglu, Å. Sivertsson, C. Kampf, E. Sjöstedt, A. Asplund, et al. "Tissue-based map of the human proteome". *Science* 347:6220, 2015, p. 1260419.

237. G. Valle-Perez, C. Q. Camargo, and A. A. Louis. "Deep learning generalizes because the parameter-function map is biased towards simple functions". In: *International Conference on Learning Representations*. 2019.

238. D. Van Dijk, R. Sharma, J. Nainys, K. Yim, P. Kathail, A. J. Carr, C. Burdziak, K. R. Moon, C. L. Chaffer, D. Pattabiraman, et al. "Recovering gene interactions from single-cell data using data diffusion". *Cell* 174:3, 2018, pp. 716–729.

239. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017, pp. 5998–6008.

240. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. "Graph Attention Networks". In: *International Conference on Learning Representations*. 2018.

241. J. Venna and S. Kaski. "Visualizing gene interaction graphs with local multidimensional scaling". In: *Proceedings of the 14th European Symposium on Artificial Neural Networks*. d-side publishing, 2006, pp. 557–562.

242. K. Vervier, P. Mahé, J.-B. Veyrieras, and J.-P. Vert. "Benchmark of structured machine learning methods for microbial identification from mass-spectrometry data". Preprint. 2015. arXiv: `1506.07251 [stat.ML]`.

243. L. Vietoris. "Über den höheren Zusammenhang kompakter Räume und eine Klasse von zusammenhangstreuen Abbildungen". *Mathematische Annalen* 97:1, 1927, pp. 454–472.

244. J. Vincent, R. Moreno, J. Takala, S. Willatts, A. De Mendonça, H. Bruining, C. Reinhart, P. Suter, and L. Thijs. "The SOFA (Sepsis-related Organ Failure Assessment) score to describe organ dysfunction/failure. On behalf of the Working Group on Sepsis-Related Problems of the European Society of Intensive Care Medicine." *Intensive Care Medicine* 22:7, 1996, pp. 707–710.

245. O. Vinyals, S. Bengio, and M. Kudlur. "Order matters: Sequence to sequence for sets". In: *International Conference on Learning Representations*. 2016.

246. K. N. Vokinger, S. Feuerriegel, and A. S. Kesselheim. “Mitigating bias in machine learning for medicine”. *Communications medicine* 1:1, 2021, pp. 1–3.

247. E. Wagstaff, F. Fuchs, M. Engelcke, I. Posner, and M. A. Osborne. “On the Limitations of Representing Functions on Sets”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 6487–6494.

248. Y. Wald, A. Feder, D. Greenfeld, and U. Shalit. “On Calibration and Out-of-Domain Generalization”. In: *Thirty-Fifth Conference on Neural Information Processing Systems*. 2021.

249. H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, W. Li, X. Xie, and M. Guo. “Learning graph representation with generative adversarial nets”. *IEEE Transactions on Knowledge and Data Engineering* 33:8, 2019, pp. 3090–3103.

250. D. J. Watts and S. H. Strogatz. “Collective dynamics of ‘small-world’ networks”. *Nature* 393:6684, 1998, pp. 440–442.

251. C. Weis, A. Cuénod, B. Rieck, O. Dubuis, S. Graf, C. Lang, M. Oberle, M. Brackmann, K. K. Søgaard, M. Osthoff, et al. “Direct antimicrobial resistance prediction from clinical MALDI-TOF mass spectra using machine learning”. *Nature Medicine* 28:1, 2022, pp. 164–174.

252. C. V. Weis, C. R. Jutzeler, and K. Borgwardt. “Machine learning for microbial identification and antimicrobial susceptibility testing on MALDI-TOF mass spectra: a systematic review”. *Clinical Microbiology and Infection*, 2020.

253. C. Weis*, M. Horn*, B. Rieck*, A. Cuénod, A. Egli, and K. Borgwardt. “Topological and kernel-based microbial phenotype prediction from MALDI-TOF mass spectra”. *Bioinformatics* 36, 2020, pp. i30–i38.

254. B. Weisfeiler and A. Leman. “The reduction of a graph to canonical form and the algebra which appears therein”. *NTI, Series* 2:9, 1968, pp. 12–16.

255. J. E. Wennberg. “Unwarranted variations in healthcare delivery: implications for academic medical centres”. *Bmj* 325:7370, 2002, pp. 961–964.

256. O. Wieder, S. Kohlbacher, M. Kuenemann, A. Garon, P. Ducrot, T. Seidel, and T. Langer. “A compact review of molecular property prediction with graph neural networks”. *Drug Discovery Today: Technologies* 37, 2020, pp. 1–12.

257. S. Wiegreffe and Y. Pinter. “Attention is not not Explanation”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 11–20.

258. A. Wong, E. Otles, J. P. Donnelly, A. Krumm, J. McCullough, O. DeTroyer-Cooley, J. Pestrue, M. Phillips, J. Konye, C. Penoza, et al. “External Validation of a Widely Implemented Proprietary Sepsis Prediction Model in Hospitalized Patients”. *JAMA Internal Medicine*, 2021.

259. Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. “A Comprehensive Survey on Graph Neural Networks”. *IEEE Transactions on Neural Networks and Learning Systems* 32:1, 2021, pp. 4–24.

260. H. Xiao, K. Rasul, and R. Vollgraf. “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms”. 2017. arXiv: cs.LG/1708.07747 [cs.LG].

261. K. Xu, W. Hu, J. Leskovec, and S. Jegelka. “How Powerful are Graph Neural Networks?” In: *International Conference on Learning Representations*. 2019.

262. K. Xu, J. Li, M. Zhang, S. S. Du, K.-i. Kawarabayashi, and S. Jegelka. “What Can Neural Networks Reason About?” In: *International Conference on Learning Representations*. 2019.

263. P. Yanardag and S. Vishwanathan. “Deep graph kernels”. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 1365–1374.

264. J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec. “Graph convolutional policy network for goal-directed molecular graph generation”. *Advances in neural information processing systems* 31, 2018.

265. J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec. “GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 5708–5717.

266. C. Yun, S. Bhojanapalli, A. S. Rawat, S. Reddi, and S. Kumar. “Are Transformers universal approximators of sequence-to-sequence functions?” In: *International Conference on Learning Representations*. 2020.

267. M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. “Deep Sets”. *Advances in Neural Information Processing Systems* 30, 2017.

268. Z. Zeng, A. K. H. Tung, J. Wang, J. Feng, and L. Zhou. “Comparing Stars: On Approximating Graph Edit Distance”. *Proceedings of the VLDB Endowment* 2:1, 2009, pp. 25–36.

269. A. Zhang, L. Xing, J. Zou, and J. C. Wu. "Shifting machine learning for healthcare from development to deployment and from models to data". *Nature Biomedical Engineering*, 2022, pp. 1–16.

270. R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. "The unreasonable effectiveness of deep features as a perceptual metric". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

271. W. Zhang, J. Jia, Z. Liu, D. Si, L. Ma, and G. Zhang. "Circulating microRNAs as biomarkers for Sepsis secondary to pneumonia diagnosed via Sepsis 3.0". *BMC Pulmonary Medicine* 19:1, 2019, pp. 1–8.

272. Q. Zhao, Z. Ye, C. Chen, and Y. Wang. "Persistence enhanced graph neural network". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2896–2906.

273. J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. "Graph neural networks: A review of methods and applications". *AI Open* 1, 2020, pp. 57–81.