Master Thesis

Topology Preserving Contrastive Learning

Simon Schindler

January, 2024

Thesis submitted in fulfillment of the requirements
for the degree of M. Sc. in Engineering awarded by the joint program
Applied Image and Signal Processing at the
University of Applied Sciences and Paris-Lodron-University of Salzburg

Supervisor: Stefan Huber

Details

Author	Simon Schindler, B. Eng.
University	Salzburg University of Applied Sciences Paris-Lodron-University of Salzburg
Degree Program	Applied Image and Signal Processing
Title of the Thesis	Topology Preserving Contrastive Learning
Keywords	persistent topology, homology, contrastive learning, machine learning, deep learning
Supervisor	FH-Prof. Dr. Stefan Huber

Abstract

This work explores the integration of Persistent Homology, a key concept in Topological Data Analysis, into Contrastive Learning, a popular method for learning representations from large unlabeled datasets. It focuses on enhancing the quality of representations produced by this unsupervised representation learning technique and investigates the potential benefits of maintaining topological information in data representations for downstream tasks.

After introducing the basic terminology and concepts of Contrastive Learning and Persistent Homology and giving a brief overview about related work, the study is structured around a series of experiments using SimCLR, a Contrastive Learning framework with the Topological Signature Loss, a differentiable loss function based on Persistent Topology.

These experiments aim at assessing the efficacy of various approaches to incorporate the Topological Signature Loss into the representation learning process while preserving or even enhancing the quality of the learned representations. The outcomes provide insights into the effectiveness of these integrations which lead to a discussion of the implications resulting in further research questions.

Affidavit

I hereby declare that I have written the presented diploma thesis entirely on my own and that I have not used any other sources apart from those given.

Salzburg, January 6, 2024	Simon Schindler
Place, date	Signature

Contents

1. Introduction **7**

2. Background **9**
2.1. Representation Learning 9
2.1.1. Introduction 9
2.1.2. Forms of Representation Learning 12
2.1.3. Deep Learning as Representation Learning 12
2.2. Contrastive Representation Learning 13
2.2.1. General Framework 13
2.2.2. Architectures 15
2.2.3. Projection Head 17
2.2.4. Augmentations 18
2.2.5. Loss function 19
2.3. Persistent Homology 21
2.3.1. Simplices and Simplicial Complexes 22
2.3.2. Simplicial Homology 24
2.3.3. Persistence and Filtrations 27

3. Related Work **30**
3.1. Topological Autoencoders 30
3.2. Connectivity-Optimized Representation Learning via Persistent Homology 31

4. Experiments **33**
4.1. Experiment 1: Variants of Topological Signature Loss 33
4.2. Experiment 2: Impact of Representation Dimensionality 35
4.3. Experiment 3: Balance between Loss Functions 36
4.4. Experiment 4: Excluding NT-Xent Regularization 36

5. Results **38**
5.1. Experiment 1 39
5.2. Experiment 2 41
5.3. Experiment 3 41
5.4. Experiment 4 42

6. Discussion **53**

7. Conclusion **56**

List of Figures **58**

Glossary **60**

List of Tables **61**

References **62**

Appendix **65**

A. Used Software Packages **65**

1. Introduction

Motivation In the evolving landscape of machine learning, a key objective has emerged: to develop representations that are not only useful and effective but also adaptable for different downstream tasks and, to a certain degree, interpretable. This goal is particularly relevant in the realm of unsupervised learning techniques. Within this domain, Contrastive Representation Learning (CL) has risen as a promising methodology, showcasing remarkable capability in deriving robust representations from unlabeled data. Concurrently, Topological Data Analysis (TDA) has been gaining traction in the machine learning community. Recent advancements, such as those documented in [1], have successfully integrated Persistent Homology (PH) a central tool in TDA into a widely used unsupervised Machine Learning (ML) architecture: The Autoencoder (AE). This success suggests that the fusion of CL, another unsupervised ML framework, with PH might yield similarly beneficial outcomes. The primary motivation is to explore whether maintaining the topological information of data in learned representations can enhance their efficacy when used in downstream tasks, thereby contributing to the advancement of unsupervised learning methodologies.

Thesis Structure This thesis is structured as follows:

- **Section 2: Theory** - Lays the foundational concepts of representation learning, with a focus on deep learning as a tool for representation learning. It also introduces CL, detailing its general framework, architectures, projection head, augmentations, and loss functions. Additionally, this section explores the concept of persistent homology, including simplices, simplicial complexes, and persistence.
- **Section 3: Related Work** - Reviews existing literature on topologically regularized AEs, situating this thesis within the broader context of current research.
- **Section 4: Experiments** - Describes the experimental approach taken in this thesis, focusing on the application of the topological signature loss in CL, particularly in the SimCLR framework. This section is subdivided into different experiments, each exploring various aspects and configurations of the application of the topological signature loss function.
- **Section 5: Results** - Presents a detailed analysis of the observations and findings from each of the experiments conducted, providing insights into the effects of topological regularization on CL.
- **Section 6: Discussion** - Discusses the implications of the results, reflecting on the successes, limitations, and potential areas of improvement in integrating topological concepts into CL.

- **Section 7: Conclusion** - Summarizes the main contributions of the thesis, revisits the findings from each experiment, and proposes directions for future research in this area of study.
- **Appendices and References** - Includes additional supporting information and citations of all references used throughout the thesis.

This structure aims to provide a comprehensive exploration of the intersection between TDA and CL, offering new perspectives and methodologies in the field of ML.

2. Background

In this section, some basic terminology and definitions are introduced to ensure the investigations in the following chapters are accessible to readers who may not have a technical background in the fields of ML and/or algebraic topology. The first subsection focuses on the concept of Representation Learning (RL). This is followed by a subsection about CL, a field of machine learning architectures used for RL. Lastly, PH, a subfield of algebraic topology, is explained in the final part of this chapter.

2.1. Representation Learning

This section provides the reader with a brief introduction to RL, and is followed by section 2.2 about CL, the RL technique used in this work.

2.1.1. Introduction

The representation of information plays a central role in the field of computer science: For example, can the efficiency at problem-solving tasks be significantly increased if the data is structured in a specifically organized format, commonly referred to as a suitable data structure. Looking at the field of machine learning the efficacy of models depends largely on the representation chosen for its input data. In the best case, an optimal representation allows all important features required for the decision-making or regression process to be incorporated into the corresponding model, while at the same time ignoring superfluous details. Conversely, in the worst case, the most interesting features of the data may not be usable for the model due to the wrong form of representation or may not stand out clearly from all the features embedded in the representation, resulting in significant underperformance.

Figure 1 illustrates this problem. It shows data of two classes in two different forms of representation, of which only in the second form the two classes would be fully separable by a linear decision boundary. A rather "simple" linear classifier like a perceptron would perform poorly if the input data was represented in cartesian coordinates, while it could solve the task with an accuracy 100% with the data in polar coordinate format.

Put simply, a good representation is one that sufficiently summarizes and highlights the most important concepts and details in the data and therefore makes a subsequent learning task easier for a "simple" model [2]. A rather broad definition of a good representation is given by Bengio et al. in [3], where they identify the following core principles:

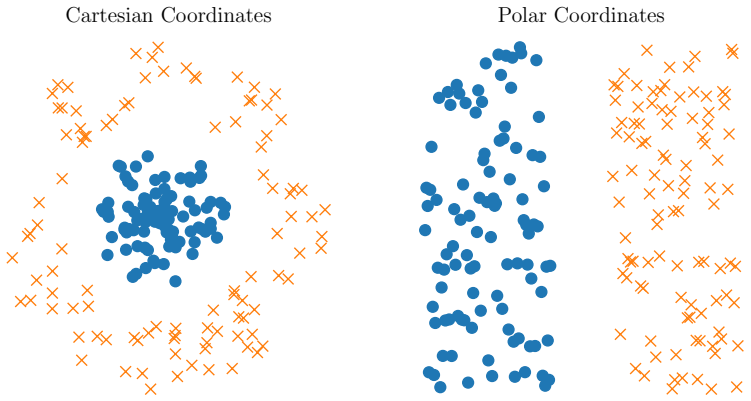

Figure 1: The same underlying data presented in cartesian and polar coordinates.

- *Hierarchical and meaningful organization of multiple explanatory factors:* This principle underscores the necessity of structuring multiple explanatory factors in a hierarchical manner, where more abstract and important concepts occupy higher positions in the hierarchy. This enhances both interpretability and robustness to minor local variations in the input space.
- *Local smoothness:* The representation should exhibit smooth transitions[1] as the input data undergoes gradual changes. This attribute proves crucial for proficiently interpolating between two samples in the representation space. Additionally, it contributes to greater invariance to localized alterations in the input data.
- *Temporal/spatial coherence:* Following from the local smoothness criterion, temporally or spatially proximate data samples should differ only slightly in their representation, since they can mostly be related to the same categorical concepts. This characteristic naturally fosters clustering within the representation space.
- *Disentanglement of concepts:* Important concepts present in the data should be partitioned in the representation space. This benefits to a higher explainability/interpretability of the representation.
- *Sparsity:* The input data should be explainable by a limited subset of latent variables[2]. While only a small fraction of these variables should be activated, the majority should remain neutral (i.e., equal to 0).

[1] In particular, the mapping f to the representation space should be Lipschitz continuous: Besides beeing continuous, the rate of change should be bound by a real number $L \geq 0$ such that the following inequality holds: $|f(x) - f(y)| \leq L|x - y|$.

[2] This assumes the manifold hypothesis stating that high dimensional real world datasets often lie along lower dimensional latent manifolds.

- *Simplicity of factor interdependencies:* If some factors are interdependent despite disentanglement, their relationship should be simple, typically linear.
- *Reusability across tasks:* The representation should be sufficient for different learning tasks and not tailored to a specific one.

It is important to note that achieving all of these properties may introduce conflicts with optimizing performance for a specific task. Consequently, the representation may lack task-specificity, potentially resulting in suboptimal performance compared to a representation fine-tuned specifically for that task, often referred to as "overfitting" in some contexts [2].

Besides being useful for downstream machine learning tasks, a good representation of data can also enable the study of inherent properties present in the data itself, without the need for a particular task.

The conventional way that has been mainly used to find good representations of some data for a machine learning task is called feature engineering [2]. It aims at generating meaningful and augmentation-invariant features from the data, depending on the model and the task. It frequently requires a substantial degree of domain expert knowledge of the data and its underlying generative mechanisms, often leading to a significant engineering endeavor. For instance, the search for optimal representations of image data in the context of machine learning spanned several decades of intensive research, a pursuit that to some extent became redundant with the advent of deep learning within the domain of computer vision.

A more automated and data-driven methodology for the creation of features is RL. Instead of relying on manually engineered features extracted from the data, valuable representations can be directly learned from the data itself through this approach. This circumvents the necessity for labor-intensive manual feature engineering while mitigating the associated potential challenges and pitfalls. Formally it aims at finding a parametric mapping $f_\Theta : X \to Y$ with $X \subset \mathbb{R}^N, Y \subset \mathbb{R}^M$ operating on the data sample $x_i \in X$ to create a suitable representation $y_i \in Y$ that captures the relevant concepts and information present in the data.
The input data in $X \subset \mathbb{R}^N$ is often high dimensional, e.g., images or videos, while the relevant information can be represented in $Y \subset \mathbb{R}^M$ with far less dimensions: $M < N$. Therefore, RL must reduce the dimensionality of the data similarly to other dimensionality reduction methods, while also learning a mapping that can be generalized to unseen data.
A major downside of RL when compared to feature engineering can be a lower interpretability of the learned representation for humans.

2.1.2. Forms of Representation Learning

Depending on the task and the respectively available data, RL can be carried out either in a supervised, unsupervised or self-supervised fashion:

- *Supervised:* The input data includes labels that are used to train a particular model for a specific inference task. The model, such as a Multi-layer Perceptron (MLP) classifier, learns an intermediate representation that is especially effective for the task at hand. In the case of a MLP every hidden layer corresponds to a intermediate representation of the data, especially useful for the subsequent layer.
- *Unsupervised:* These methods are independent of the existence of labels for the data set used. Representations are derived by analyzing relationships between samples in the input data, e.g., with clustering algorithms or Principal Component Analysis (PCA).
- *Self-supervised:* Label pairs are constructed for each sample in the unlabeled input data. In this way, supervised training methods can be used even if there are no labels, e.g., with CL introduced in section 2.2 or AEs.

2.1.3. Deep Learning as Representation Learning

The Deep Learning (DL) paradigm can be considered as a form of layered representation learning. It has shown remarkable effectiveness in a wide range of applications in the recent years, mainly due to its ability to capture nested, hierarchical and complex representations and the associated mappings adapted to specific tasks.
DL encompasses capabilities beyond the mere acquisition of task-specific representations through optimization. Specifically designed architectures such as Convolutional Neural Networks (CNNs), Long Short Term Memorys (LSTMs) and transformer networks aswell as techniques such as regularization, pruning, and early stopping enhance the quality of the learned representations, aligning them more with the criteria articulated by [3] and cross-referenced in section 2.1.1:

- *Hierarchical and meaningful organization of multiple explanatory factors:* Within the context of neural networks, each layer is designed to construct a representation of input data that possesses utility for subsequent layers, ultimately culminating in a representation tailored to the specific task at hand.
- *Local smoothness:* Local smoothness is typically maintained by restricting the transformations between layers in the neural network to continuous mappings.
- *Temporal and spatial coherence:* Temporal coherence is a characteristic of LSTMs and transformer networks, while spatial coherence is a feature of CNNs.

- *Disentanglement of concepts:* The pursuit of disentangled representation learning is exemplified by approaches like the β - Variational Auto Encoder (β-VAE).
- *Sparsity:* Achieving sparsity in representations is attainable through the implementation of appropriate regularization and pruning techniques.
- *Reusability across tasks:* The concept of reusability across diverse tasks, facilitated by Transfer Learning, enables the utilization of a common model base and the associated representations for addressing different tasks.

2.2. Contrastive Representation Learning

Although DL is frequently used within supervised learning settings, there is a notable and growing interest in the utilization of self-supervised and unsupervised methodologies. The reason for this lies in the potential to operate without the need for labeled data, as unlabeled data resources are often abundant. Conversely, the process of data labeling is resource-intensive and susceptible to inaccuracies and biases introduced by humans, thereby potentially compromising the quality of learned representations and through this the overall performance of the models.

Abundant, unlabeled data, therefore can be used in the acquisition of basic representations, which subsequently may undergo fine-tuning with labeled data to fit for specific tasks.

A very general approach and learning paradigm that makes use of the inherent information within extensive unlabeled datasets for the generation of high-quality representations is known as CL. The core idea behind it is to teach a model to distinguish between instances of data points with the ultimate goal to learn generalizable features in a task-agnostic manner. Should the model acquire the ability to differentiate between distinct instances without incorporating knowledge about categories, it may develop a representation that primarily captures apparent similarity among these instances. This is analogous to how class-wise supervised learning also preserves perceptible similarities within different classes.

2.2.1. General Framework

This subsection gives a short overview about the general building blocks of a CL architecture and the nomenclature.

In the CL framework positive pairs denote pairs of data points x_i, x_i' created by two distinct functions for augmentation $t_i, t_i' \in T$ from a set of possible augmentations T of the very same data point $x \in X$ from a dataset $X \subset \mathbb{R}^N$. Therefore the pair is considered similar and both augmentations share membership in the same pseudo class. For example, in computer vision, two distinct augmentations of a single image may constitute a positive pair.

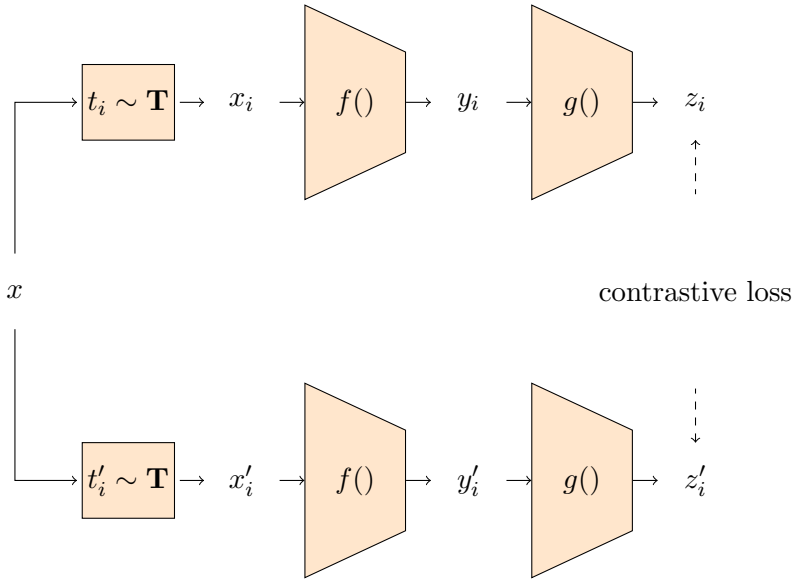

Figure 2: The general contrastive learning framework consisting of an input sample $x \in X$ two different augmentations $t_i, t'_i \in T$, the augmented versions of the image $x_i, x'_i \in \mathbb{R}^N$ the encoder network $f : \mathbb{R}^N \to \mathbb{R}^M$ the representations $y_i, y'_i \in \mathbb{R}^M$ the projection head $g : \mathbb{R}^M \to \mathbb{R}^K$ and the embeddings $z_i, z'_i \in \mathbb{R}^K$ on which the contrastive loss is then calculated.

Conversely, negative pairs are constructed by pairing augmented data points perceived as dissimilar. These negative pairs can involve, for instance, combining an augmented image with an entirely unrelated image drawn from the dataset.

All pairs are fed through a parameterizable function $f : \mathbb{R}^N \to \mathbb{R}^M$ that is usually a neural network creating the so called representations y_i, y'_i. The data is then projected by f into a usually lower dimensional space $Y \subset \mathbb{R}^M$, known as the representation space.

In some architectures the computed representations are subsequently reduced in dimensionality by another parameterizable function $g : \mathbb{R}^M \to \mathbb{R}^K$, called the projection head, which is usually a shallow neural network, into an embedding space $Z \subset \mathbb{R}^K$ wherein the similarity of the pair is calculated. Other architectures, that do not apply a projection head, compute the similarity directly on the representations in Y. For a more in depth description of the projection head resort to section 2.2.3.

To train the model, a contrastive loss function is employed as the optimization objective to maximize a given similarity metric between the embeddings of positive pairs, utilizing metrics such as the euclidean distance or the cosine similarity, while at the same time maximizing the separation between the embeddings of negative pairs.

Throughout the training process, the model acquires the ability to draw the embeddings of positive pairs closer together and push the embeddings of negative pairs further apart. This iterative refinement improves the model's capacity to capture meaningful information in its representations and drives the model to become invariant to the augmentations in T.

A rather formal description of the goal of CL would be the following:
Consider the set of all possible embeddings Z, which represents the union of all possible augmentations subsequently projected by the encoder network f and the projection head g. This can be formally expressed as:

$$Z = \bigcup_{t \in T} g(f(t(X)))$$

Next, define a relation $\equiv$ on Z such that two elements u and v are related (i.e., $u \equiv v$) if and only if there exist transformations t, t' in T and an input $x \in X$ for which $u = g(f(t(x)))$ and $v = g(f(t'(x)))$.

The equivalence classes formed by this relation $\equiv$ are the sets of augmentations $t(f(g(X)))$ for each transformation t in T. These classes partition the set Z.

The objective of CL is to bring these equivalence classes closer together by minimizing the contrastive loss, effectively contracting the space of the augmentations within each class.

Informally the CL framework can be viewed as a dictionary lookup. In this analogy, the encoded input data acts as the query, and the encoded samples serve as the keys in the dictionary. For a positive pair a query and a key are two augmented versions of the same source image, which should be very similar in the representation space, while it should not for negative pairs.

2.2.2. Architectures

In their 2020 survey on CL Jaiswal et al. identified four main architectures into which strategies for CL can be clustered. The following will briefly introduce all four of them and their respective advantages and drawbacks.

End-to-End: The end-to-end learning system represents the most straightforward approach to CL and shares similarities with the general framework illustrated in Figure 2. In alignment with the general framework, the End-to-End system incorporates two encoders, specifically a query encoder denoted as f_q and a key encoder referred to as f_k. Importantly, these two encoders may be the same but can be also distinct from each other, and they undergo simultaneous end-to-end updates via backpropagation during the training process.

With the exception of the two augmented versions of the initial image x_i, x_i', all other augmentations within the batch are treated as negatives. It is worth noting that the number of negative samples in this approach corresponds to the batch size, as it aggregates negative samples from the current batch. Consequently, the framework particularly benefits from larger batch sizes. However, it is crucial to recognize that the batch size is restricted by the available memory, posing an ongoing challenge for scalability in these methods.

The most prominent representative of this approach is SimCLR introduced by Chen et al. in [4], where batch sizes up to 4096 were employed, and the projection head was introduced to enhance performance, with f_q being set equal to f_k. SimCLR frequently serves as a benchmark for evaluating other CL architectures.

Memory Bank: The concept of maintaining a detached set of representations for the entire dataset with the purpose of instance discrimination was introduced by Wu et al. in 2018 in [5]. This approach allowed the incorporation of a large number of negative samples without increasing the batch to an infeasibly large size. The stored representations within this memory bank are subsequently chosen randomly to function as the keys (the negatives in the CL loss), while the queries are generated by the query encoder f_q from two distinct transformations of the same images. The representations produced by the query encoder are then saved in the memory bank for further utilization as keys in the following epoch. Since the negative samples used for the loss calculation are detached, the learning objective is just optimized with respect to the the representations of the current mini batch of every step.

One potential drawback of this approach is the computational cost of maintaining current representations within the memory bank, as they tend to quickly become outdated within a few iterations. This occurs because the key encoder, responsible for generating representations for each mini-batch, undergoes changes with each optimization step, potentially leading to inconsistent representations, which may not be useful when comparing to more recently generated representations.

Momentum Encoder: In their work to MoCov1 [6] He et al. introduced the idea of using a dynamically changing encoder for the creation and maintenance of a dynamic dictionary. Based on their hypothesis that "good features can be learned by a large dictionary that covers a rich set of negative samples, while the encoder for the dictionary keys is kept as consistent as possible despite its evolution" they introduce a modification of the memory bank architecture by substituting the conventional memory bank with a concept known as a "momentum encoder", which facilitates the creation of a dynamic dictionary. In this dynamic dictionary approach, each new batch of training data is initially processed by the momentum (key) encoder denoted as f_k and subsequently added to a queue within the dictionary. This dictionary queue is then employed for the selection of negative samples. When the queue size reaches a predefined threshold, the oldest batch is dequeued to maintain the queue's size within the limit. The formation of positive pairs involves encoding one augmentation, x_i, using the query encoder f_q, and encoding the other augmentation, x_i', using the key

encoder f_k. To maintain the stability and consistency of the dictionary, where the keys are derived from a slowly evolving encoder, the key encoder f_k is designed as a momentum-based moving average of the query encoder f_q. Following an update to f_q through backpropagation with a contrastive loss function in conjunction with an optimizer, the parameters θ_k of f_k are adjusted according to the following equation:

$$\theta_k = m\theta_k + (1 - m)\theta_q \tag{1}$$

In [6] a relatively large momentum like $m = 0.999$ yielded better results than smaller values like $m = 0.9$. In contrast to the End-to-End framework, where the dictionary size equals the size of the current mini batch, this architecture allows for arbitrary dictionary sizes. This comes with the benefit of less memory consumption in the training stage, since smaller batch sizes are also effective, while maintaining similar or even better performance. For example MoCov2 [7], which is a slightly adopted version of MoCov1 [6], utilizing a projection head in the training stage, yielded better performance than SimCLR [4] (71.1 % vs 69.3 %) when trained and evaluated with a linear classifier on the imagenet dataset while using a much smaller batch size for training (256 vs. 4096).

Clustering: One problem with the previously discussed methods lies in their formal treatment of each instance as its own unique class. This approach tends to separate instances that should belong to the same class while just drawing similar augmentations of the same instance closer together. This outcome is undesirable since images with similar semantics should ideally result in similar embeddings.

Caron et al. try to resolve this issue of CL with their SwAV architecture [8] by utilizing a clustering algorithm to not only draw a pair of samples close to each other as in above approaches, but to also ensure that all other representations that are similar to each other form clusters together. In order to achieve this, they allocate the representations of two distinct augmentations of an image, denoted as y_i and y_i', to their nearest clusters c_a and c_b selected from a finite set $\{c_1, ..., c_k\}$. In the loss computation, these assignments are interchanged, such that y_i is associated with c_b and y_i' with c_a. This, in essence, is designed to ultimately ensure that y_i and y_i' are attributed to the same clusters and enables augmentations of other instances that are similar in the representation space to also be assigned to that cluster. Compared to the above mentioned contrastive methods, this approach demonstrates enhanced memory efficiency, as it doesn't need a memory bank nor a dictionary queue and also works well with smaller batch sizes than the End-to-End approach [8].

2.2.3. Projection Head

Using a nonlinear projection head g, i.e. a shallow MLP in combination with an activation layer, to project the representations y_i and y_i' into a lower dimensional embedding space Z while training does significantly improve the quality of the representations. This, in return, yields a notable improvement in performance across downstream

tasks, as established by prior research [4, 9]. In the work by T. Chen et al. [4], an over 10 percent enhancement in performance was documented, while X. Chen et al. [9] reported a gain of over 5 percent in classification accuracy when employing a non-linear projection head as opposed to directly using the representation for the loss calculation.

Gupta et al. try to answer the question why such a projection head improves the representation quality in such a significant way in [10]. Following their training of the widely recognized CL model, SimCLR, as introduced in the work by Chen et al. [4], Gupta et al. conducted an analysis of the vector spaces into which both the encoder h and the projection head g map their respective inputs. Notably, the projection head g was investigated in three configurations: (1) as an identity mapping, (2) as a single-layer perceptron and (3) as a MLP. In the assessment of the numerical rank[3] for both the projection head output space Z and the encoder output space Y, through `torch.linalg.matrix_rank`, an interesting trajectory was revealed: starting from (1) the absence of a projection head, progressing to (2) a linear projection, and finally to (3) a non-linear projection, there was a notable escalation in the rank of Y, coupled with a reduction in the rank of Z. Additionally, they observed a significant improvement in performance on downstream tasks as they increased the complexity, and consequently, the learning capacity of the projection head g from (1) to (2) to (3). They emphasize this as a positive correlation between the rank deficit between Y and Z and performance in downstream tasks. Building on these empirical findings, they propose a hypothesis suggesting that the projection head implicitly acquires the ability to select a subset of features for the application of the contrastive loss. This, in turn, allows the projection head's output to effectively minimize the contrastive loss, while affording the encoder f the flexibility to learn features that are more generalizable. The contrastive loss, such as the NT-Xent employed in this work, is formulated with the goal of eliminating the impact of data augmentations while retaining the underlying content information within the embedding it is employed on. By utilizing the projection head, it becomes possible to achieve this objective, all the while permitting the retention of data augmentation information within the representation space Y, which might be important for downstream tasks.

2.2.4. Augmentations

Data augmentation plays a crucial role in CL by diversifying the training data, and consequently enhancing model robustness, improving its capacity for feature discrimination, and mitigating issues related to overfitting. The employment of various transformations as part of the augmentation process yields a more extensive and diverse

[3] The numerical matrix rank of a space A corresponds to the number of significant singular values in the covariance matrix of the representations in that space, produced by the model when processing test data.

array of positive and negative data pairs, thereby forcing the model to learn augmentation invariant features which enhances its capability to generalize effectively when confronted with previously unseen data. This stands in contrast to supervised learning. While, generally speaking, data augmentations present a useful regularization technique for supervised learning that helps to prevent overfitting, overly aggressive data augmentations can actually have an adverse effect on the model's performance [3].

The specific arrangement and choice of data transforms plays a critical role for the quality of representations learned by the model.

Chen et al. experimented in their original work on SimCLR [4], to determine the most effective combination and order of transformations to benefit the quality of the learned representation. The findings of this investigation revealed that while individual transformations were adequate to enable the model to differentiate between positive and negative pairs (i.e. minimize the contrastive loss), they were insufficient to promote the learning of high-quality representations. The set of augmentation techniques examined contains both spatial/geometric transformations (such as cropping, resizing, flipping, rotating, and region removal) and appearance transformations (including color distortion, Gaussian blurring, and Sobel filtering). An illustration for transformations can be seen in Figure 3. The incorporation of multiple transformations in sequence introduced a higher level of complexity into the contrastive prediction task, resulting in a substantial enhancement of the acquired representations. Notably, the combination of random cropping (along with resizing) and color distortion emerged as the most effective in improving the model's performance in downstream tasks. Chen et al. argue that this particular combination, featuring both a spatial and an appearance transformation, yields augmentation pairs with dissimilar characteristics in terms of spatial content and color distribution. This divergence might force the model to learn more abstract and fundamental concepts inherent in the underlying data.

2.2.5. Loss function

Contrastive losses are employed to enhance the similarity between representations of augmented pairs of data samples. As opposed to having explicit target labels, the loss relies on the notion of positive and negative pairs, where positive pairs should have similar representations, and negative pairs have dissimilar ones.

A commonly used loss is the NT-Xent, the normalized temperature-scaled cross entropy loss. Formally the loss for one positive pair of embeddings (z_i, z_i') can be expressed as follows:

Definition 1 (NT-Xent)

$$L_i = -\log \frac{\exp(\text{sim}(z_i, z_i')/\tau)}{\sum_{j \neq i} \exp(\text{sim}(z_i, z_j)/\tau)}$$

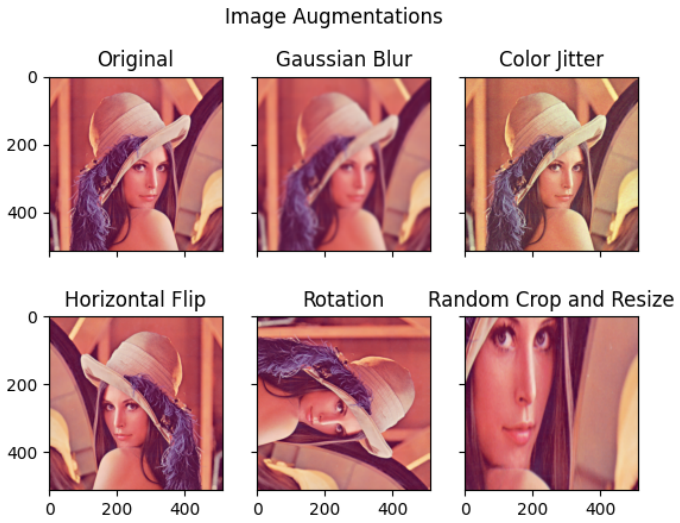

Figure 3: Different augmentations applied to an image.

Here, $\text{sim}(z_i, z_i')$ represents the similarity measure between the representations z_i and z_i' e.g. using the cosine similarity function $\text{sim}(u, v) = \frac{\langle u,v \rangle}{\|u\|\|v\|}$. The temperature parameter τ scales the logits, influencing the level of discrimination between positive and negative pairs during training. This can lead to enhanced robustness to variations in the input space and aid in stabilizing the training process [11]. There is a total number of $2N$ augmentations constructed from a batch of input data of size N. For every sample of those $2N$ augmentations there exists one positive sample and $2(N-1)$ negative ones, resulting in N positive pairs per batch. The total contrastive loss L_c is calculated over all N positive pairs:

$$L_c = \frac{1}{N} \sum_{i=1}^{N} L_i = \frac{1}{N} \sum_{i=1}^{N} (-\text{sim}(z_i, z_i')/\tau + \log \sum_{j \neq i} \exp(\text{sim}(z_i, z_j)/\tau)) \tag{2}$$

2.3. Persistent Homology

The field of TDA builds on the idea that the topological structure of a data set can be useful for inference of information about the underlying data generating process. It is a suited tool for the analysis of high-dimensional data and has been successfully applied in a number of fields such as computer vision [12], medical imaging [13] and computational biology [14].

TDA builds on the field of algebraic topology, a branch of mathematics dealing with qualitative geometric information using abstract algebra. Algebraic topology is devoted to the precise formalization of connectivity information, such as the classification of connected components, loops and higher dimensional surfaces in space.

A concept rooted in algebraic topology is simplicial homology, an approach to quantitatively and unambiguously formalize the existence of multidimensional holes contained in spaces, so-called simplicial complexes. These topological properties are encoded by the Betti numbers of the spaces, which represent the numbers of k-dimensional holes in the space ($k \in \mathbb{N}_0$).

PH is a technique of the field of TDA that applies homology for examination of the topological properties within the data. This is done by first converting the data into a suitable representation and then monitoring how the set of k-dimensional holes within the data evolves across different scales of observation, a so-called filtration. In other words the *persistence* of the topological features (the homology classes) within the data across different scales is analyzed. It is motivated by the assumption, that persistent features are more meaningful than spurious ones.

Recent publications such as [1, 15, 16] have combined PH with ML to fulfill topological constraints, such as the preservation of the homology of the input data, after the application of an auto encoder network [1].

Given that the concept of PH is not widely recognized within the ML community, yet has been gaining interest in recent years due to its potential as a valuable tool for ML engineers and researchers, this work is designed to provide a brief introduction to the fundamental principles of PH. The goal is to equip the reader with the necessary understanding for the subsequent chapters of this work.

The basic concepts of PH are outlined in the following section. Starting with the definition of simplicial complexes in subsection 2.3.1 basics of simplicial homology are presented in subsection 2.3.2. Building on those foundations the key concepts of PH are explained in subsection 2.3.3.

The structure and form of presentation is based on and inspired by the following works: [17–21].

2.3.1. Simplices and Simplicial Complexes

A way of representation of a topological space is its decomposition into simple pieces, such as the triangulation of points in a plane, where the whole triangulation as a topological space can be broken down into triangles. Such a decomposition is called a complex given its pieces are topological simple and their common intersections are lower dimensional complexes of the same kind [18]. A so-called simplicial complex is a structure, that is employed to represent the topological space. It can be further broken down into basic topologically simple components known as simplices [4]. The following definitions are necessary for the formulation of a simplicial complex:

Definition 2 (Affine combination) *An affine combination of $k+1$ points $u_0, \ldots, u_k \in \mathbb{R}^d$ is $x = \sum_{i=0}^{k} \lambda_i u_i$ if $\sum_{i=0}^{k} \lambda_i = 1$ with $\lambda_i \in \mathbb{R}$.*

Definition 3 (Convex combination) *A convex combination of $k+1$ points $u_0, \ldots, u_k \in \mathbb{R}^d$ $x = \sum_{i=0}^{k} \lambda_i u_i$ is an affine combination with $\lambda_i \geq 0$.*

Definition 4 (Affine independence) *The $k+1$ points $u_0, \ldots, u_k$ are affinely independent if there is no affine combination of them that is equal to the zero vector.*

Definition 5 (Convex hull) *The convex hull $conv\{u_0, \ldots, u_k\}$ of $k+1$ points $u_0, \ldots, u_k \in \mathbb{R}^d$ is the set of all convex combinations of these points.*

Definition 6 (Simplex) *A k-simplex σ is the convex hull $conv\{u_0, \ldots, u_k\}$ of $k+1$ affinely independent points $u_0, \ldots, u_k$.*

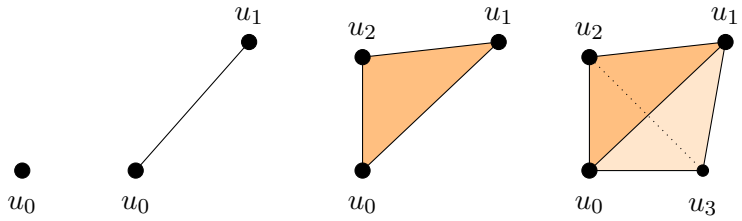

Figure 4: A 0-, a 1-, a 2- and a 3-simplex.

In the geometric interpretation, a k-simplex is in general the simplest possible convex polytope in a given dimension k.

In the following $\sigma = (u_0, \ldots, u_k)$ is expressing that the simplex σ is spanned by the points $u_0, \ldots, u_k$.

[4] singular: "simplex"

For example a 3-simplex as shown on the right in Figure 4 is the convex hull of four affinely independent points in $\mathbb{R}^3$: It can be decomposed into four 2-simplices (faces), these into six 1-simplices (edges) and these into four 0-simplices (vertices).

A face τ of a simplex $\sigma = conv\{u_0, \ldots, u_k\}$ is a simplex itself. It is the convex hull of a non-empty subset of $\{u_0, \ldots, u_k\}$ and is proper if the subset is not the entire set.

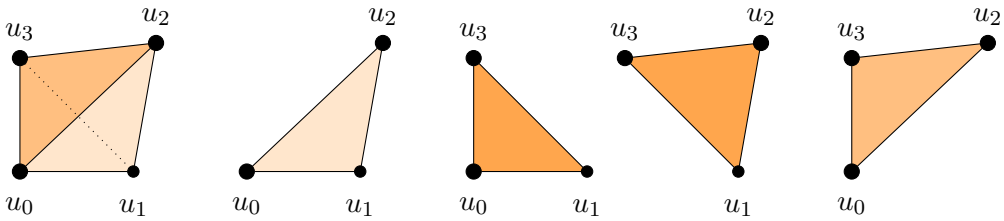

Figure 5: A 3-simplex and its 2-dimensional faces.

Figure 5 shows all four 2-dimensional faces of a 3-simplex as an example. All of these are proper faces. The 3-simplex itself would be an improper face.

Definition 7 (Simplicial complex) *A simplicial complex K is a finite set of simplices σ if $\sigma \in K$ implies that $\tau \in K$ for every face τ of σ, and if $\sigma_1 \cap \sigma_2 \in K$ for every non-disjoint pair $\sigma_1, \sigma_2 \in K$.*

In simple words a simplicial complex is a set of simplices that is closed under taking faces and intersections of the simplices [17]. Any two simplices are either disjoint or meet in a common face. The dimension of a simplicial complex is the maximum dimension of any of its simplices. The simplicial complex is then referred to as a k-simplicial complex.

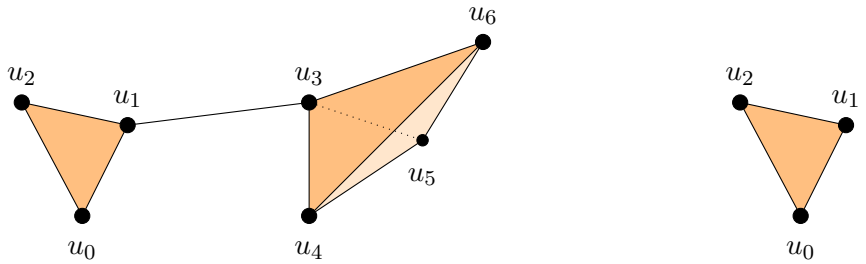

Figure 6: A 3-simplicial complex on the left and a 2-subcomplex on the right.

Definition 8 (Subcomplex) *A subcomplex K' of a simplicial complex K is a simplicial complex such that $K' \subset K$.*

Figure 6 shows a 3-dimensional simplicial complex and one 2-dimensional subcomplex for an example.

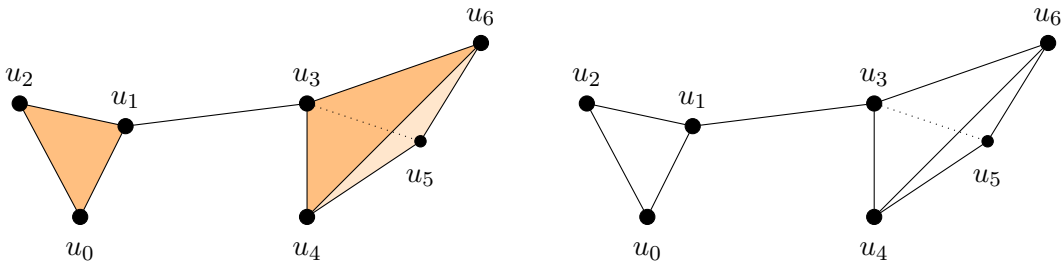

Figure 7: A 3-simplicial complex on the left its 1-skeleton on the right.

Definition 9 (j-skeleton) *A j-skeleton $K^{(j)}$ of a simplicial complex K is the subcomplex of K that consists of all simplices in K of dimension at most j. $K^{(j)} = \{\sigma \in K | dim\sigma \leq j\}$.*

Figure 7 shows a 3-simplicial complex and its 1-skeleton consisting of 0- and 1-simplices (points and edges).

Considering the geometric realization of a simplicial complex is not always necessary, and in many cases, the abstract structure alone suffices and proves to be more practical. Hence, the subsequent definition of an abstract simplicial complex is provided:

Definition 10 (Abstract simplicial complex) *An abstract simplicial complex A is a finite collection of sets such that for every $X \in A$ and $Y \subseteq X$ we have $Y \in A$.*

2.3.2. Simplicial Homology

Simplicial homology is a way of quantitatively describing the topological structure of a space building on the concept of simplices and simplicial complexes as introduced in subsection 2.3.1. It aims to identify the topological features of the space X, namely the number of p-dimensional holes, quantified by the so called p-th Betti number. The underlying concept necessary for the computation of the Betti numbers are the homology groups of a simplicial complex.

The following definitions are needed for the formal definition of homology groups and their corresponding Betti numbers at the end of this section.

Definition 11 (p-chain) *A p-chain $c = \sum \alpha_i \sigma_i$ of a simplicial complex K is a formal sum of simplices σ_i of dimension p in K with coefficients $\alpha_i \in \mathbb{Z}_2$.*

The coefficients α_i count the modulo 2 multiplicity of the presence of the corresponding simplex in the p-chain. They are either 0 or 1, because a simplex can either be part of a chain or not. Figure 8 shows a 2-simplex. A 1-chain in this example would be $c_1 = a + b$.

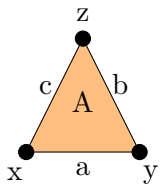

Figure 8: A 2-simplex.

Group	Generating Set	Rank
C_0	$\{x, y, z\}$	3
C_1	$\{a, b, c\}$	3
C_2	$\{A\}$	1
C_3	$\{0\}$	0
$Z_0 = ker\partial_0 : C_0 \to 0$	$\{x, y, z\}$	3
$Z_1 = ker\partial_1 : C_1 \to C_0$	$\{a + b + c\}$	1
$Z_2 = ker\partial_2 : C_2 \to C_1$	$\{0\}$	0
$B_0 = im\partial_1 : C_1 \to C_0$	$\{x + y, y + z, z + x\}$	2
$B_1 = im\partial_2 : C_2 \to C_1$	$\{a + b + c\}$	1
$B_2 = im\partial_2 : C_3 \to C_2$	$\{0\}$	0

Table 1: The groups of the simplex in Figure 8.

The addition of two chains $c_1 = \sum \alpha_i \sigma_i, c_2 = \sum \beta_i \sigma_i$ is defined as the sum of the coefficients of the chains:

Definition 12 (Addition of p-chains) $c_1 + c_2 = \sum \alpha_i \sigma_i + \sum \beta_i \sigma_i = \sum (\alpha_i + \beta_i)\sigma_i$ *with* $\alpha_i, \beta_i \in \mathbb{Z}_2$

The set of p-chains in a complex K together with the addition defined in Def. 12 forms an abelian group because of the associativity and commutativity of the addition, the existence of the identity element $(c + 0 = c)$ and the existence of the inverse element $(c + c = 0)$.

Definition 13 (Group of p-chains) *The group of p-chains is denoted by $(C_p, +)$ or $C_p(K)$ formed by the set of p-chains in K with the defined addition (Def. 12).*

The boundary operator ∂_p defines the boundaries of a p-simplex $\sigma = (u_0, \ldots, u_p)$ as follows:

Definition 14 (Boundary operator ∂_p) $\partial_p(\sigma) = \sum_{i=0}^{p} (u_0, \ldots, \hat{u_i}, \ldots, u_p)$, *where $\hat{u_i}$ means that u_i is not part of the simplex.*

Definition 15 (Boundary of a p-chain) *The boundary of a chain is defined as the sum of the boundaries of the simplices in the chain:* $\partial_p(c) = \sum_i \alpha_i \partial_p(\sigma_i)$

As an example, consider the 3-simplex in Figure 8. The boundary of the edge b is $\partial_1(b) = y + z$. The boundary of the face A is $\partial_2(A) = a + b + c$.

The boundary operator is a map of p-chains to $(p-1)$-chains $\partial_p : C_p \to C_{p-1}$ and preserves the defined addition (Def. 12) $\partial_p(c_1 + c_2) = \partial_p(c_1) + \partial_p(c_2)$. Therefore it is a homomorphism called the p-th boundary homomorphism.

Definition 16 (p-cycle) *A p-cycle c is a p-chain that has no boundary, i.e.,* $\partial_p(c) = 0$.

Definition 17 (Set of p-cycles) *The set of p-cycles is denoted by Z_p. It is a subset of the p-chains C_p, $Z_p \subseteq C_p$. Since every p-cycle $c \in Z_p$ fulfills $\partial_p(c) = 0$ the set of the p-cycles Z_p is the kernel (nullspace) of the boundary operator ∂_p:* $Z_p = \ker \partial_p$.

Another important subset of C_p is the set of p-boundaries B_p representing the boundaries of the $p+1$-chains in C_{p+1}:

Definition 18 (Set of p-boundaries) *The set of p-boundaries is denoted by B_p. It is a subset of the p-chains C_p, $B_p \subseteq C_p$. Since every p-boundary $b \in B_p$ encloses a $(p+1)$-simplex, the set of the p-boundaries B_p is the image of the boundary operator ∂_{p+1}:* $B_p = \operatorname{Im} \partial_{p+1}$.

By definition, a p-boundary encloses a $(p+1)$-simplex in the complex and is therefore a p-cycle. Because of that, the set of p-boundaries is a subset of the set of p-cycles, which itself is a subset of the set of p-chains: $B_p \subseteq Z_p \subseteq C_p$.

Since the boundary of a p-cycle is always 0 and every p-boundary is a p-cycle the following holds:

Lemma 1 $\partial_p(\partial_{p+1}(\sigma_{p+1})) = 0$

These definitions suffice for the following definition of the p-th homology group mentioned in the very beginning of this section:

Definition 19 (p-th homology group) *The p-th homology group is the group of equivalence classes (homology classes) formed by the quotient group of the group of p-cycles by the set of p-boundaries:* $H_p = Z_p / B_p = \ker \partial_p / \operatorname{Im} \partial_{p+1}$

Definition 20 (Betti number) *The p-th Betti number is the rank of the p-th homology group:* $\beta_p = \operatorname{rank}(H_p) = \operatorname{rank}(\ker \partial_p / \operatorname{Im} \partial_{p+1}) = \operatorname{rank}(Z_p) - \operatorname{rank}(B_p)$

The Betti numbers are quite intuitive. They represent the number of p-dimensional holes in the complex. If a p-hole gets filled by a $p+1$-simplex the p-boundary b_p that encloses a $p+1$-simplex in the complex is added to the generator set of the group of boundaries B_p leading to a higher of B_p and therefore reducing the rank of the group H_p (the p-th Betti number) by one.

Definition 21 (Chain complex) *A chain complex is assembled from a set of p-chain groups together with a set of boundary maps:*

$$\ldots \xrightarrow{\partial_{p+1}} C_p \xrightarrow{\partial_p} C_{p-1} \xrightarrow{\partial_{p-1}} \ldots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} 0$$

2.3.3. Persistence and Filtrations

Raw point cloud data usually does not come in the form of a simplicial complex. Therefore, homology without any adaption would not yield any useful information from the data. This lack of connection between the data points is addressed by PH. Through the stepwise construction of a simplicial complex from the data, connections in the form of p-simplices between the data points are established and topological relations can be inferred. This procedure results in a filtration, a sequence of nested simplicial complexes starting with an empty complex and ending in the complex K itself.

Definition 22 (filtration) *A filtration of a simplicial complex K is a sequence of nested simplicial complexes $\emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n = K$.*

This procedure can be thought of as a stepwise insertion of simplices to the complex K. It depends on a chosen filtration function $f : K \to \mathbb{R}$, which assigns each simplex $\sigma \in K$ a real number that describes its time of appearance in the filtration. The filtration function f must be non-decreasing along the increasing chain of faces, i.e. $f(\sigma) \leq f(\tau)$ for all simplices $\sigma, \tau \in K$ with $\sigma \subseteq \tau$.

The Vietoris-Rips complex, originally introduced by Leopold Vietoris [22], is an abstract simplicial complex frequently employed in the computation of PH for point cloud data. It serves as a mathematical representation that captures topological features within the point cloud data at a particular spatial scale.

Definition 23 (Vietoris-Rips complex) *Given a specified scale parameter ϵ the Vietoris-Rips complex $\mathcal{R}_\epsilon(\mathbf{X})$ of a point cloud dataset $\mathbf{X}$ contains all simplices derived from $\mathbf{X}$ whose elements $\{x_0, x_1, \ldots\}$ satisfy the condition $dist(x_i, x_j) \leq \epsilon$ for all combinations of i and j, where $dist : \mathbf{R}^n \to \mathbf{R}$ is a distance metric of choice.*

By increasing the scale parameter ϵ a filtration of nested Vietoris-Rips complexes can be produced since $\mathcal{R}_{\epsilon_1}(\mathbf{X}) \subseteq \mathcal{R}_{\epsilon_2}(\mathbf{X})$ if $\epsilon_1 < \epsilon_2$.

Let σ_i be the p-simplex that is inserted into the complex K_i to form the complex K_{i+1} and $a_i = f(\sigma_i)$ the corresponding filtration value. The insertion of σ_i can affect either the homology group $H_p(K_{i+1})$ or $H_{p-1}(K_{i+1})$. In the first case σ_i closes a p-dimensional cycle and the generator set of the group of cycles $Z_p(K_{i+1})$ is extended by one element, whereas in the second case σ_i fills a p-dimensional hole resulting in a reduction of the generator set of $B_{p-1}(K_{i+1})$ by one element.

Since $K_i \subseteq K_j$ for $i \leq j$ the inclusion induces a homomorphism $f_p^{i,j} : H_p(K_i) \to H_p(K_j)$ for every dimension p. The filtration results in a sequence of homology groups for each dimension p, which elements are mapped to each other by the homomorphism: $0 = H_p(K_0) \to H_p(K_1) \to \cdots \to H_p(K_n) = H_p(K)$.

Let $C_p^i \subseteq K_i$ be the group of p-chains, Z_p^i the group of p-cycles and B_p^i the group of p-boundaries in the complex K_i at filtration step i. The persistent p-th homology group is hence defined as:

Definition 24 (persistent p-th homology group) $H_p^{i,j}(K) = \operatorname{Im} f_p^{i,j} = Z_p^i/(B_p^j \cap Z_p^i)$ *for* $0 \leq i \leq j \leq n$.

Its generator set consists of all p-cycles that are in the image of the homomorphism $f_p^{i,j}$. These are the p-cycles that come to exist latest in the filtration step i and are filled earliest in the filtration step j. The persistent p-th homology group is hence a measure of the topological properties of the data that are stable over the filtration process from step i to step j.

With the persistent homology groups the persistent Betti numbers can be defined analogously:

Definition 25 (persistent p-th Betti number) $b_p^{i,j}(K) = \operatorname{rank}(H_p^{i,j}(K))$ *for* $i \leq j$.

They measure the number of p-dimensional holes that persist over the filtration process from step i to step j.

Definition 26 (persistence of a homology class) *The persistence of a homology class* γ *that is born into* K_i *and dies in* K_j *is defined as:* $\operatorname{pers}(\gamma) = f(\sigma_j) - f(\sigma_i) = a_j - a_i$.

Similarly the index persistence of a class γ is defined as:

Definition 27 (index persistence of a homology class) $\operatorname{ipers}(\gamma) = j - i$.

After computation of the persistent homology groups and the persistent Betti numbers the so-called p-th persistence diagram $D_p(f)$ can be constructed for every dimensionality p. It is a multiset of points in the extended real plane $\bar{\mathbb{R}}^2 = (\mathbb{R} \cup \{-\infty, +\infty\})^2$, and often displayed as a 2-dimensional plot, where the x-axis represents the time of birth of a homology class in the filtration and the y-axis the time of death. Each point in the diagram represents a homology class that is born in the filtration step i and dies in the filtration step j. The vertical distance between the point and the diagonal line $x = y$ is the persistence of the class.

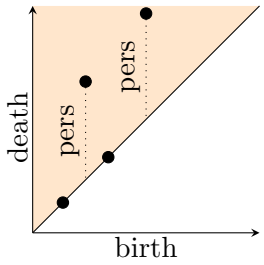

Figure 9: An example persistence diagram displaying the time of birth and time of death of four homology classes.

Figure 9 shows an example of a persistence diagram. Entries of homology classes will generally be in the region above the diagonal since they can not disappear from the filtration before they appear.

The points also carry information about the multiplicity $\mu_p^{i,j}$ of the class, i.e. the number of homology classes that are born in the filtration step i and die in the filtration step j. The multiplicity $\mu_p^{i,j}$ can be computed from the persistent Betti numbers:

Definition 28 (multiplicity of a persistent homology class) $\mu_p^{i,j} = (b_p^{i,j-1} - b_p^{i,j}) - (b_p^{i-1,j-1} - b_p^{i-1,j})$.

Homology classes that are born and die in the same filtration lie on the diagonal and have infinite multiplicity out of technical reasons (see [18] for further details).

Lemma 2 (Fundamental lemma of PH) *For a filtration of a simplicial complex* $\emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n = K$ *and every pair of indices* (k, l) *with* $0 \leq k \leq l \leq n$ *the* p*-th persistent Betti number is:* $b_p^{k,l} = \sum_{i \leq k} \sum_{j \geq l} \mu_p^{i,j}$

The lemma states that the persistent Betti numbers can be computed just by summing over some multiplicity values in the multiset $\{\mu_p^{i,j}\}$ with $0 \leq i \leq j \leq n$ that is encoded in the persistence diagram, implying that all information about the persistence in the filtration is encoded in the persistence diagram. This property makes the persistence diagrams a suitable topological information source for machine learning algorithms.

3. Related Work

This chapter reviews two research papers, which focus on the integration of PH into unsupervised RL. These works provide a theoretical and practical basis for the approaches explored in this thesis.

3.1. Topological Autoencoders

In order to maintain the topological relationships among data points within a dataset while simultaneously decreasing its dimensionality, Moore et al. employ an AE architecture for dimensionality reduction in [1] and formulate a differentiable topological loss term grounded in PH.

To introduce this loss, it is imperative to establish several formal definitions in advance:

By computing all pairwise distances between elements in a given point cloud X and organizing them into a distance matrix A^X (where $A^X_{ij} = d(x_i, x_j)$), and by utilizing a matrix π_d for every dimensionality d monitoring the indices i and j of the simplices σ_i and σ_j in the Vietoris-Rips complex $\mathcal{R}_{\tilde{\epsilon}}(X)$ that create or destroy a topological feature, denoted as a point $(a, b) \in D_d$ in the persistence diagram, it is possible to directly reconstruct D_d using the information from both A^X and π_d. Selecting the distances from A^X that correspond to the indices in π_d yields a $n \times 2$ matrix filled with the coordinates of the n topological features in the d-dimensional persistence diagram D_d. This matrix is denoted as $A^X[\pi_d]$ in the following.

The paper refers to (D^X, π^X) as the result of a PH algorithm on the filtration of the Vietoris-Rips complex $\mathcal{R}_{\tilde{\epsilon}}(X)$. Where D^X consists of the persistence diagrams D_d in various dimensions d and π^X of the corresponding persistence pairs π_d.

As illustrated in Figure 10 an AE can be seen as a composition of two mappings $f \circ g$, usually neural networks, where $f : \mathbb{R}^m \to \mathbb{R}^n$ is the so called encoder that maps from the input data space $X \subset \mathbb{R}^m$ to the latent representation space $Z \subset \mathbb{R}^n$ with $n << m$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ is the decoder that maps back to the reconstruction space $X' \subset \mathbb{R}^m$.

Definition 29 (Topological Signature Loss) $L_T = L_{X \to Z} + L_{Z \to X}$
with $L_{X \to Z} = \frac{1}{2} \| A^X[\pi^X] - A^Z[\pi^X] \|^2$
and $L_{Z \to X} = \frac{1}{2} \| A^Z[\pi^Z] - A^X[\pi^Z] \|^2$

The topological loss term aims at harmonising the topology between a batch of input data $X = \{x_1 \dots x_k\} \subset \mathbb{R}^m$ and its the latent representations $Z = f(X) = \{z_1 \dots z_k\} \subset \mathbb{R}^n$ by comparing their d-dimensional persistence diagrams D_d. These are generated from the filtration of the Vietoris-Rips complexes $\mathcal{R}_{\tilde{\epsilon}}(X)$ and $\mathcal{R}_{\tilde{\epsilon}}(Z)$,

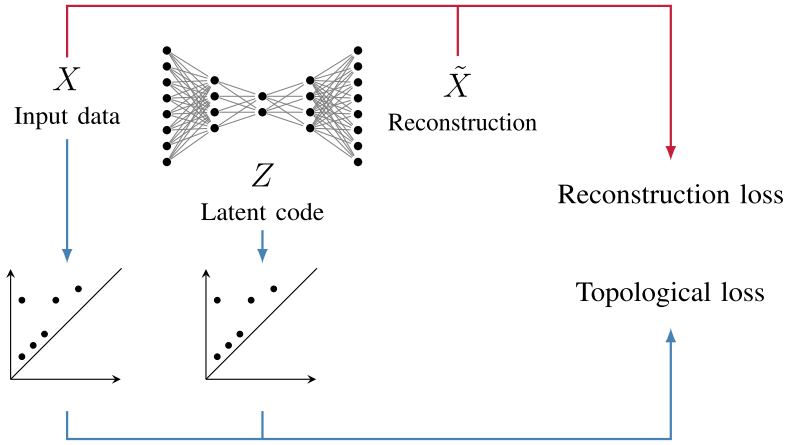

Figure 10: An overview of the method presented by Moore et al. in [1]. The graphic was taken from the original paper.

with $\widetilde{\epsilon}$ representing the scale parameter at which the connectivity within $\mathcal{R}_{\widetilde{\epsilon}}(X)$ or $\mathcal{R}_{\widetilde{\epsilon}}(Z)$ reaches a stable state. Note that such a stable state exists for all finite data sets such as X and Z.

3.2. Connectivity-Optimized Representation Learning via Persistent Homology

Similar to the Topological Autoencoder by Moore et al. in [1] described in the last subsection, Hofer et al. [15] regularize an AEs latent space via a loss function relying on information from PH.

Even though both works build on the AE architecture with the goal to learn useful representations for downstream tasks incorporating PH in the process, there is a major difference between both approaches: Instead of preserving the connectivity information from the input space X in the representation space Y the loss function described in this publication, termed the *connectivity loss*, solely acts on the latent space.

The following definitions are necessary for the formal introduction of the connectivity loss:

Definition 30 (Death Times) *With $S \subset \mathbb{R}^n$ being a finite set, $(\epsilon_k)_{k=1}^M$ being the increasing sequence of pairwise distance values of S and $D_0(S)$ being the 0-dimensional*

persistence diagram of S, $\dagger(S)$ can be defined as the multi-set of death-times

$$\dagger(S) = \{t : (0, \epsilon_t/2) \in D_0(S)\}$$

with t contained in $\dagger(S)$ with the same multiplicity as $(0, \epsilon_t/2)$ in $D_0(S)$.

Definition 31 (Connectivity Loss) *The connectivity loss $L_\eta(S)$ for a hyperparameter $\eta \in \mathbb{R}^+$ can then be defined:*

$$L_\eta(S) = \sum_{t \in \dagger(S)} |\eta - \epsilon_t|$$

It regularizes the times of death of 0-dimensional homology classes, i.e. the pairwise distances of the 0-simplices dying at the filtration step t being incorporated into a 1-simplicial complex in the vietoris rips filtration $\mathcal{R}_{\epsilon_t}(S)$ of a pointcloud S.

This loss was then successfully applied to the representation Y of a AE for One Class Classification (OCC).

4. Experiments

Motivated by the successful combination of the Topological Signature Loss (Definition 29) and the AE architecture in [1] this work aims at investigating if and how this loss could also be used to enhance the quality of representations produced by a CL architecture. For this purpose the first experiment evaluates different points of application of the Topological Signature Loss within a prominent CL architecture. The most promising variant is then further explored in the following experiments.

While more advanced models exist as elaborated in the theory section to CL (Section 2.2) a rather moderately complex model, the End-to-End framework SimCLR [4], is used for all of the following experiments with the goal to ensure interpretability.

For a list of the used software packages and a link to the implementations refer to the Appendix A.

The dataset employed for the following experiments is CIFAR-10 [23]. It is widely used in machine learning and consists of 60,000 color images of size 32x32 in 10 different classes. These classes include common objects such as airplanes, automobiles, birds, cats, and more. Given that GPU memory is limited to 48 GB, the small image size of CIFAR-10 is particularly advantageous for this use case, enabling the use of larger batch sizes during training. This, in turn, enhances the quality of learned representations, as demonstrated by SimCLR [4]. Furthermore, Moore et al. also used CIFAR-10 for the evaluation of their topologically regularized autoencoders in [1].

4.1. Experiment 1: Variants of Topological Signature Loss

This experiment exhaustively explores possible applications of the Topological Signature Loss within the SimCLR architecture with the goal to identify the most promising for further investigations. The End-to-End framework of SimCLR allows for multiple variants of application of the Signature Loss:

- **Variant 1:** Harmonizing the topology between the augmented batches of input data X_i, X_i' and the projection heads outputs $Z_i = g(f(X_i))$, $Z_i' = g(f(X_i'))$.
  $L_{T1} = L_{X_i \to Z_i} + L_{Z_i \to X_i} + L_{X_i' \to Z_i'} + L_{Z_i' \to X_i'}$
- **Variant 2:** Harmonizing the topology between the augmented batches of input data X_i, X_i' and their representations $Y_i = f(X_i)$, $Y_i' = f(X_i')$.
  $L_{T2} = L_{X_i \to Y_i} + L_{Y_i \to X_i} + L_{X_i' \to Y_i'} + L_{Y_i' \to X_i'}$
- **Variant 3:** Harmonizing the topology between the representations $Y_i = f(X_i)$, $Y_i' = f(X_i')$ and the projection heads outputs $Z_i = g(f(X_i))$, $Z_i' = g(f(X_i'))$.
  $L_{T3} = L_{Y_i \to Z_i} + L_{Z_i \to Y_i} + L_{Y_i' \to Z_i'} + L_{Z_i' \to Y_i'}$

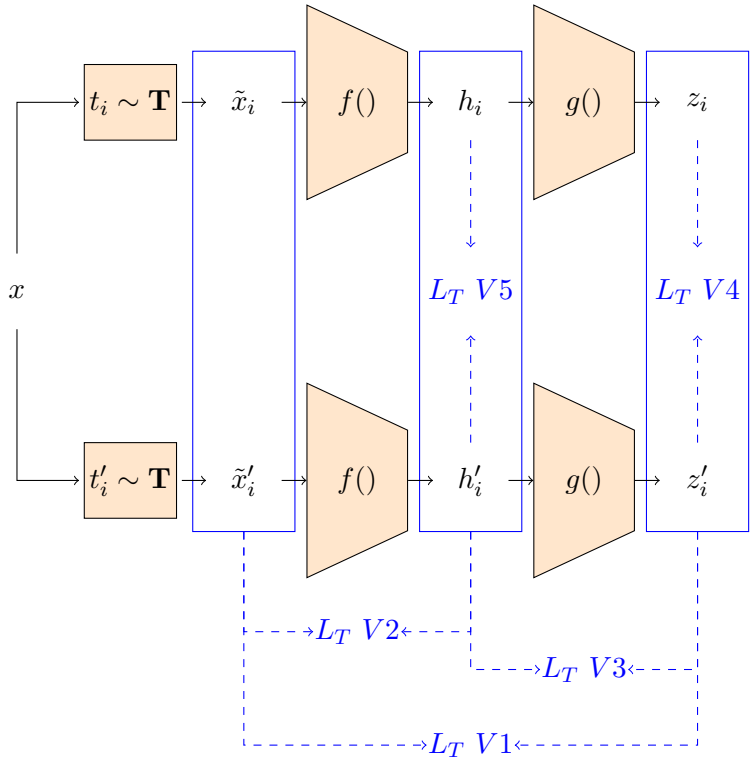

Figure 11: Variants of topological signature loss calculation in Experiment 1

- **Variant 4:** Harmonizing the topology between the projection heads output batches $Z_i = g(f(X_i))$ and $Z'_i = g(f(X'_i))$.
  $L_{T4} = L_{Z'_i \to Z_i} + L_{Z_i \to Z'_i}$
- **Variant 5:** Harmonizing the topology between the representation batches $Y_i = f(X_i)$ and $Y'_i = f(X'_i)$.
  $L_{T5} = L_{Y'_i \to Y_i} + L_{Y_i \to Y'_i}$

Figure 11 illustrates the application points of Variants 1-5 within the SimCLR architecture.

Similar to Moore et al. in [1], Variants 1 and 2 aim to preserve the inherent topology of the input space in the embedding space. This preservation could potentially be advantageous for downstream tasks, especially if topological information is crucial for solving a particular problem. Variant 3 can be viewed as a regularization of the projection head, ensuring that no topological information is discarded before the calculation of the contrastive loss. Variants 4 and 5 encourage functions f and g to topologically align the pairs of embedded batches of data. This alignments, similar to the contrastive loss, may enhance invariance to the augmentations applied to the input data, since both the "upper" and the "lower" augmentations should become similar topology

wise and cosine similarity wise even though different strong augmentations have been applied to both sides.

To ensure that meaningful representations are learned during training, the total loss term consists of one of the five variants for the signature loss L_{Tx} with $x \in [1, 2, 3, 4, 5]$ scaled by the hyperparameter $\lambda \in \mathbf{R}_+$ and a contrastive loss L_c, in this case, the NT-Xent loss (Definition 1), applied to the output of the projection head:

$$L = L_c + \lambda L_{Tx}$$

A spectrum of values $[0, 1, 10, 100, 1000]$ for the hyperparameter λ is employed to evaluate the impact of the signature loss on the training dynamics, as well as on the resultant learned representations and projections. The training is stopped after 100 epochs per configuration, a choice based on prior empirical observations that the learning curve typically plateaus after approximately 100 epochs, coupled with resource constraints. The Adam optimizer [24] is selected, with a learning rate set to 10^{-3} and a weight decay of 10^{-6}. Due to hardware limitations the batch size is limited to 512. The temperature τ choosen for the NT-Xent loss is 0.5.

Parameter	**Value**
Batch size	[8, 64, 512]
Epochs	100
Learning rate	1×10^{-3}
Weight decay	1×10^{-6}
Optimizer	Adam
λ	[0, 1, 10, 100, 1000]
τ	0.5

Table 2: Hyperparameters for model training in Experiment 1

4.2. Experiment 2: Impact of Representation Dimensionality

In the course of assessing the models derived from Experiment 1 (Subsection 4.1), a hypothesis emerged that suggested that Variant 5 of the application of the Topological Signature Loss substantially enhances the model's capability of embedding relevant information for downstream tasks, utilizing a reduced number of significant principal components. To empirically validate this hypothesis, the dimensionality N of the representation space $Y \subset \mathbb{R}^N$, is set in this experiment via the incorporation of a singular linear layer appended to the function f. Four variants for the embedding dimensionality are employed. This slightly adapted architecture of SimCLR is then trained with and without the implementation of Variant 5 of the Topological Signature Loss exhaustively using all combinations of the hyperparameters listed in Table 3.

Parameter	Value
Batch size	[8, 64, 512]
Epochs	100
Learning rate	1×10^{-3}
Weight decay	1×10^{-6}
Optimizer	Adam
λ	[0, 100, 1000]
τ	0.5
N	[512, 1024, 1536, 2048]

Table 3: Hyperparameters for model training in Experiment 2

4.3. Experiment 3: Balance between Loss Functions

To better understand how Variant 5 of the Topological Signature Loss impacts the performance of downstream tasks that use the learned representation, and to explore if the Topological Signature Loss alone is suitable for contrastive learning, a new parameter β is added to this experiment. This leads to the overall loss being calculated in the following way:

$$L = (1 - \beta)L_c + \beta\lambda L_{Tx}$$

No additional layer is introduced to vary the representation spaces number of dimensions. λ is kept constant at $\lambda = 100$. Table 4 lists the hyperparamters used for this experiment.

Parameter	Value
Batch size	[8, 64, 512]
Epochs	100
Learning rate	1×10^{-3}
Weight decay	1×10^{-6}
Optimizer	Adam
λ	100
β	$[10^{-3}, 10^{-2}, 10^{-1}, 10^{0}]$
τ	0.5

Table 4: Hyperparameters for model training in Experiment 3

4.4. Experiment 4: Excluding NT-Xent Regularization

A recent publication by Ben-Shaul et al. [25] that sheds more light on the underlying mechanics of certain CL algorithms elaborated the importance of the regularization

term within the loss term for the successful clustering of data with respect to semantic labels.

Following their argumentation the NT-Xent loss term (Definition 1) for a positive pair of embeddings (z_i, z_i') can be viewed as an invariance and an regularization term:

$$L_i = -\log \frac{\exp(\text{sim}(z_i, z_i')/\tau)}{\sum_{j \neq i} \exp(\text{sim}(z_i, z_j')/\tau)}$$

$$= \underbrace{-\text{sim}(z_i, z_i')/\tau}_{\text{invariance}} + \underbrace{\log \sum_{j \neq i} \exp(\text{sim}(z_i, z_j')/\tau)}_{\text{regularization}}$$

The invariance term encourages the model to embed positive pairs as similar as possible while the regularization term discourages similarity of negative pairs. The contrastive loss L_c for the two corresponding batches of embeddings Z, Z' of batch size N can then be written as:

$$L_c = \frac{1}{N} \sum_{i=1}^{N} (-\text{sim}(z_i, z_i')/\tau + \log \sum_{j \neq i} \exp(\text{sim}(z_i, z_j')/\tau))$$

$$= \underbrace{-\frac{1}{N} \sum_{i=1}^{N} \text{sim}(z_i, z_i')/\tau}_{L_{cI}} + \underbrace{\frac{1}{N} \sum_{i=1}^{N} \log \sum_{j \neq i} \exp(\text{sim}(z_i, z_j')/\tau)}_{L_{cR}}$$

To evaluate the efficacy of the Topological Signature Loss for regularization within the CL framework SimCLR L_{cR} is neglected in the loss calculation for this experiment. The total loss L is then compromised as follows:

$$L = L_{cI} + \gamma L_{T5}$$

Table 5 lists the hyperparamters used for this experiment.

Parameter	**Value**
Batch size	[8, 64, 512]
Epochs	100
Learning rate	1×10^{-3}
Weight decay	1×10^{-6}
Optimizer	Adam
γ	$[10^{-1}, 10^{0}, 10^{1}, 10^{2}]$
τ	0.5

Table 5: Hyperparameters for model training in Experiment 4

5. Results

This section gives a brief description of the metrics and methods used for the evaluation of the experiments and subsequently describes the observations, useful for qualitative insights and conclusions, that can be made upon inspection of the figures.

The assessments presented in this section aid to systematically evaluate the experiments described in the last section. These assessments are crucial, to enable a deeper understanding of the impact of the signature loss, its location and the influences of the different hyperparameters in the experiments on various aspects of model training and the characteristics of representations generated by the models when applied to test data.

Loss When training neural networks with multiple loss functions, special care must be taken, since different loss functions can vary greatly in magnitude and can negatively affect each other when they are minimized together. One way to address these issues is by adjusting the balance between these loss functions using a hyperparameter, referred to as λ. In this case a higher λ leads to a higher ratio between the Topological Signature Loss and the constrastive loss. Figure 12 visualizes this ratio for all configurations while training for experiment 1.

Downstream Performance The arguably most important measure when analyzing the learned representations is the performance that can be achieved utilizing the representation for downstream tasks. To evaluate this a K-Nearest Neighbor (KNN) classifier was applied on the representations of test data after every epoch of training. Figures 13, 16, 19 and 22 show the evolution of the KNNs accuracy while training for each experiment respectively.

Figures 17, 20 and 23 show the accuracy of a 2-layer MLP classifier with sigmoid activation and layer widths of 2048 and 2048 trained on the representations generated by the various model versions after 100 epochs of training employing the various hyperparamters and variants of experiment 2, 3 and 4 respectively. The MLP was trained using the Adam optimizer [24] with the learning rate set to 10^{-3} and the weight decay set to 10^{-6}. Training was stopped after 10 epochs. Figure 14 shows the accuracy of a KNN instead of a MLP since training an MLP for every one of the 63 contrastive models in Experiment 1 would have been very computationally expensive, while the KNN classifier is trained in seconds and already gives useful insights into the potential performance of each variant.

Numerical Rank The lower subplots in the figures 14, 17, 20 and 23 visualize the numerical matrix rank of the representations, which corresponds to the number of significant singular values in the covariance matrix of the representations produced by

the models when processing test data. Here, "significant" refers to those singular values exceeding $\max(S) \times \max(M, N) \times \epsilon$, where S represents the set of singular values, M the number of representations/test samples, N the dimensionality of the representations, and ϵ the machine epsilon for the datatype of S, which is a single-precision float in this context. This calculation was performed using `numpy.linalg.matrix_rank`.

Representation Space To enhance the understanding of why representations generated by the variants of contrastive learning models used in the experiments perform in downstream tasks to a different degree, it is beneficial to examine the characteristics of the representation space more closely then just looking at the numerical matrix rank. For that reason, the eigenvalue spectrum of the covariance matrix derived from the representations obtained by applying the models to unseen test data is analyzed similar as in other work to SimCLR like [10, 26]. Additionally, the singular value spectrum of these representations when they are combined into a single $M \times N$ matrix, where M represents the number of representations or test samples, and N indicates the dimensionality of these representations is inspected. The eigenvalue and singular value spectra for each experiment and the different batch sizes 8, 64, and 512 are depicted in Figures 15, 18, 21 and 24.

5.1. Experiment 1

Downstream Task Performance An inspection of the bar chart depicting the respective KNNs performance using the representations from a model variant after beiing trained for 100 epochs in Figure 14 reveals that Variants 1 and 2 exhibit poor performance across all batch sizes and λ values, underachieving relative to the baseline. Variants 3 and 4 demonstrate moderate improvements over 1 and 2 but do not surpass the baseline. Variant 5 stands out, maintaining relatively high performance with increasing λ values, especially for batch sizes of 8 and 64, indicating potential for subsequent research.

Impact of Batch Size Larger batch sizes are correlated with improved downstream performance when training for a low number of epochs similar to this experiment, as identified in the SimCLR paper [4]. This is due to the higher number of negative examples present in the NT-Xent loss (Definition 1) per optimization step, leading to faster convergence. However, this trend does not hold for Variants 1-5. Excluding λ equal to 1 cases, where topological loss is minimal compared to contrastive loss, Variants 1 and 2 exhibit declining performance with increased batch sizes, while Variants 3, 4, and 5 peak at a batch size of 64.

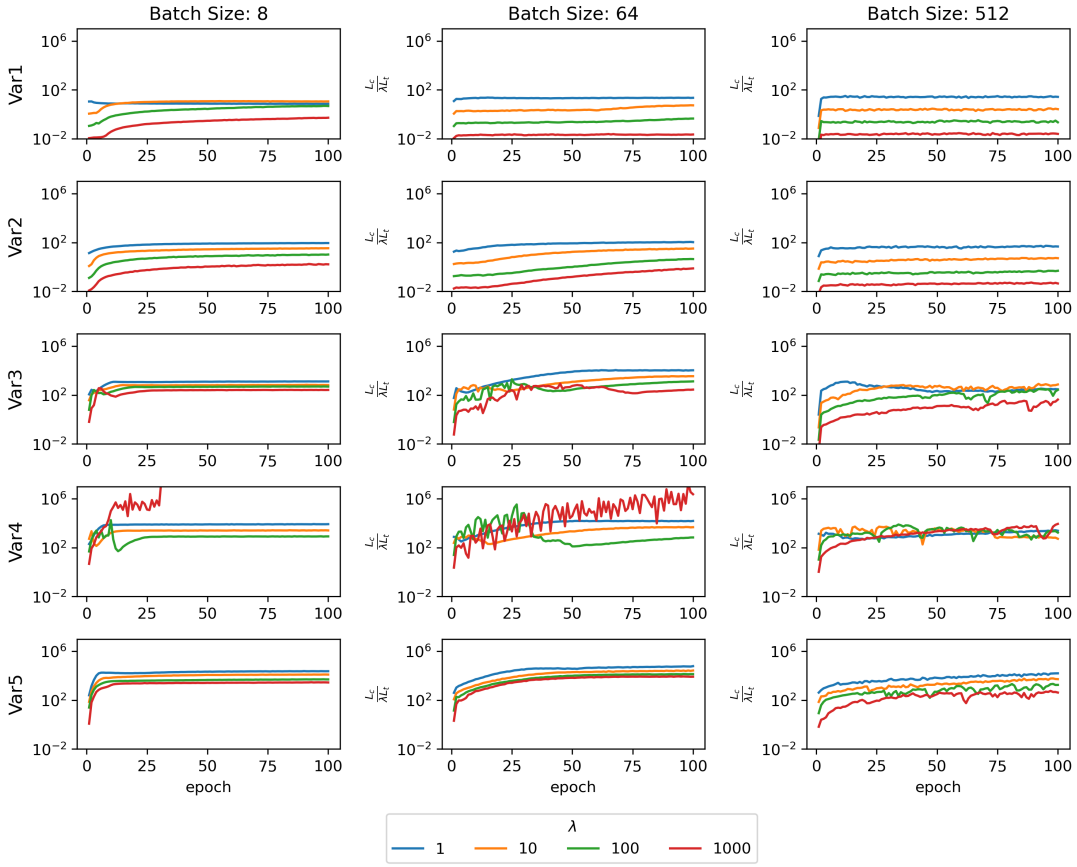

Figure 12: **Experiment 1:** Illustration of the dynamic balance between the topological signature loss and the contrastive loss during the training. Each curve represents a different configuration, showing how varying the hyperparameter λ influences the ratio of these two loss functions for a different different batch size in each column.

Lambda Variation A general trend can be observed showing a decline in downstream performance with escalating λ values. Contrarily, Variant 5 at smaller batch sizes, like 8 and 64, does not follow this pattern, where a λ of 1000 significantly boosts performance over lower λ values for batch size 8 and and just slightly reduces performance for batch size 64 but not as drastically as for Variants 1-4. It is evident that a higher λ accelerates dimensional collapse, more so with smaller batch sizes where its impact is amplified.

Representation Space The dimensional collapse intensifies with the smaller batch sizes (8 and 64), as can be seen in Figure 14, indicated by the matrix rank in the lower subplot. Variant 5 is notably quick to collapse. Figure 15 shows the eigen and singular

value spectra of Variant 5 and the baseline model. What stands out is, that at a batch size of 64 and a λ of 1000, Variant 5's eigen and singular value spectra, although lower overall, decline more gradually than for other λ values, while its downstream task performance still nearly matches the baseline despite the spectrum's lowered scale. For a batch size of 8, the spectrum at $\lambda = 1000$ not only is lower but also declines more rapidly compared to λ values of 1, 10, and 100, but still achieves comparable performance to the baseline.

5.2. Experiment 2

Training Procedure Investigation of the experiment through Figure 16 reveals that the increasement of λ leads to a delayed improvement in the downstream task performance for batch sizes 8 and 64. However, for batch size 512, Variant 5's downstream tasks performance did not improve within the first 100 epochs of training at all. Furthermore, when comparing the training curves of the same setups with different sizes of representation dimensions, no major differences are noticeable. This suggests that the performance doesn't change much with the change in representation dimensionality in these particular settings.

Downstream Task Performance In contrast to Experiment 1 Figure 17 exhibits a general trend wherein an increased level of topological regularization correlates with a decrease in downstream task performance. Notably, while models with topological regularization trained on batch sizes of 8 and 64 respectively maintained reasonably good performance when use to produce representations for the classification task, their performance was significantly inferior to the baseline model when trained with a batch size of 512.

Furthermore, the performance of the models when used for downstream tasks appears to be largely unaffected by variations in the number of representation dimensions. This could imply that the relevant information for this classification task and the chosen dataset lies on a low dimensional manifold with a dimensionality $n < 512$ in the representation space, which still can be constructed by f even when restricted to only use 512 dimensions.

Representation Space Based on the eigenvalue spectra in Figure 18, no new insights emerge from this experiment.

5.3. Experiment 3

Training Procedure Some interesting insights can be found in Figure 19 visualizing the performance of a KNN classifier for every epoch of the training procedure: For

$\beta = 1$, which implies that the optimizer exclusively relies on the Topological Signature Loss L_t without the contrastive loss L_c, the downstream performance for batch sizes 8 and 64 collapses to a level comparable to random guessing, whereas at a batch size of 512 the model only manages to preserve initial KNN performance but not to improve it. Conclusively the Topological Signature Loss alone does not suffice for contrastive learning, at least with this architecture.

Conversely, for other values of β, the learning curves are remarkably similar across batch sizes 8 and 64, with only minor variations observable. This suggests a degree of consistency in the model's learned representations across these β values in smaller batch sizes.

However, another notable divergence can be seen in the performance for $\beta = 0.1$ at a batch size of 512, which results in significantly poorer performance compared to the other settings. For the other β settings, the learning curves are again relatively similar.

Downstream Performance Similar to the findings in Experiment 2, an increase in the level of topological regularization (reflected by higher β values) correlates with a decrease in downstream task performance. Interestingly the MLP in Figure 20, for a batch size of 512 and $\beta = 1$, achieves only an accuracy equivalent to random guessing (approximately 10%). In contrast, the KNN classifier, fitted after every training epoch, consistently maintained an accuracy around 30% as illustrated in Figure 19. This disparity highlights a methodological problem in this research: simpler approaches, such as using a shallow MLP, might not be adequate for evaluating the effectiveness of the representations generated by the models. Although the KNN classifier is not particularly complex, it yields superior performance in this context. This discrepancy opens up room for further research, which could involve evaluating different types of classification algorithms on the representations produced by CL models incorporating topological regularization.

Representation Space Figure 21 visualizes that as in the previous experiments a higher degree of topological regularization in this loss configuration leads to faster collapses of the eigenvalue and singular value spectra.

5.4. Experiment 4

Training Procedure As visualized in Figure 22, in this setup, the initial performance of the KNN algorithm was roughly 30%. This performance was only sustained for a batch size of 512. However, for batch sizes of 8 and 64, the performance rapidly decreased to 10% - equivalent to random guess accuracy for this dataset - within 10 and 60 epochs, respectively. A notable observation was that a higher value of γ resulted in a slower decline in KNN performance during training.

Downstream Performance When applying a MLP to the learned representations for classifying new test data, the results were poorer than the KNN classifier, with an accuracy around 10%, mirroring the random guess accuracy for this dataset.

Representation Space The eigenvalue and singular value spectra of the covariance matrices, corresponding to the representations generated with varying γ values for batch sizes 8 and 64, showed an immediate collapse after the first index. Therefore, they are not included in Figure 24. For a batch size of 512, a pattern emerged, revealing an earlier collapse of the spectra at lower values of γ. This was the only experiment in this study that demonstrated a dimensional collapse for a batch size of 512 and Variant 5 of the Topological Signature Loss function.

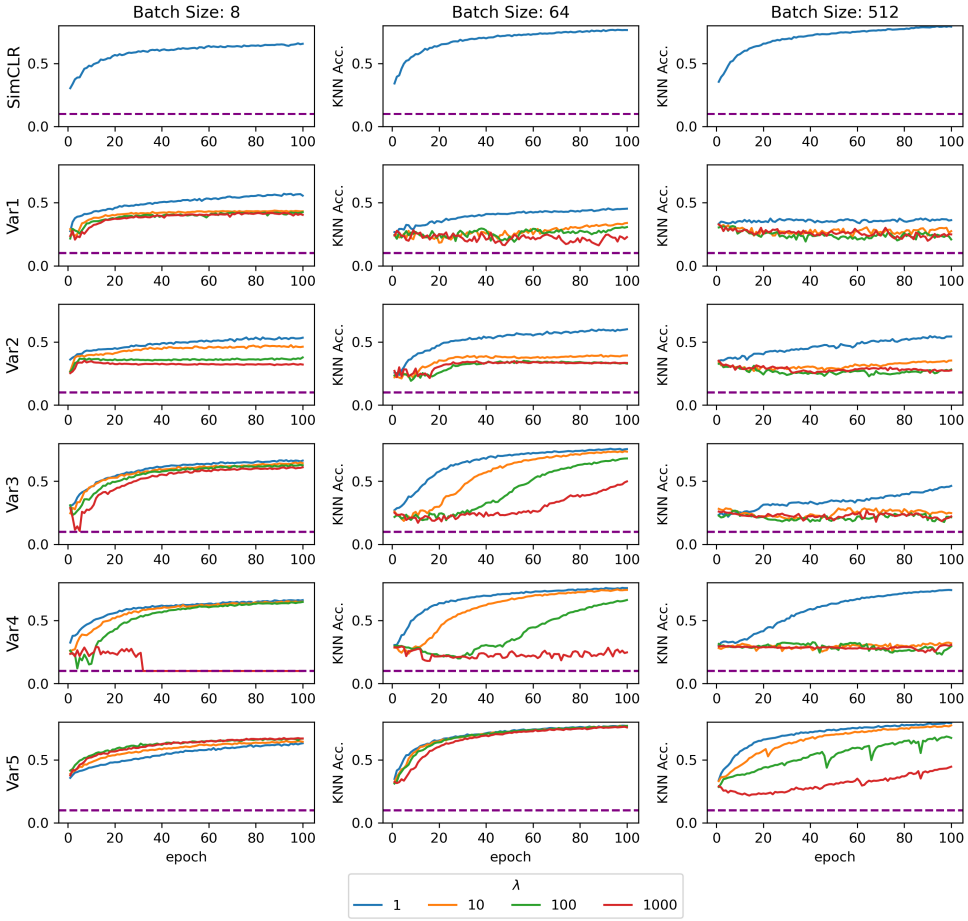

Figure 13: **Experiment 1:** Illustration of the progression of a KNN classifiers accuracy over each training epoch, applied to the representations of test data for different training batch sizes and all variants of the loss calculation with different values for λ, as well as the baseline model. The purple line marks the accuracy for random guessing (10%).

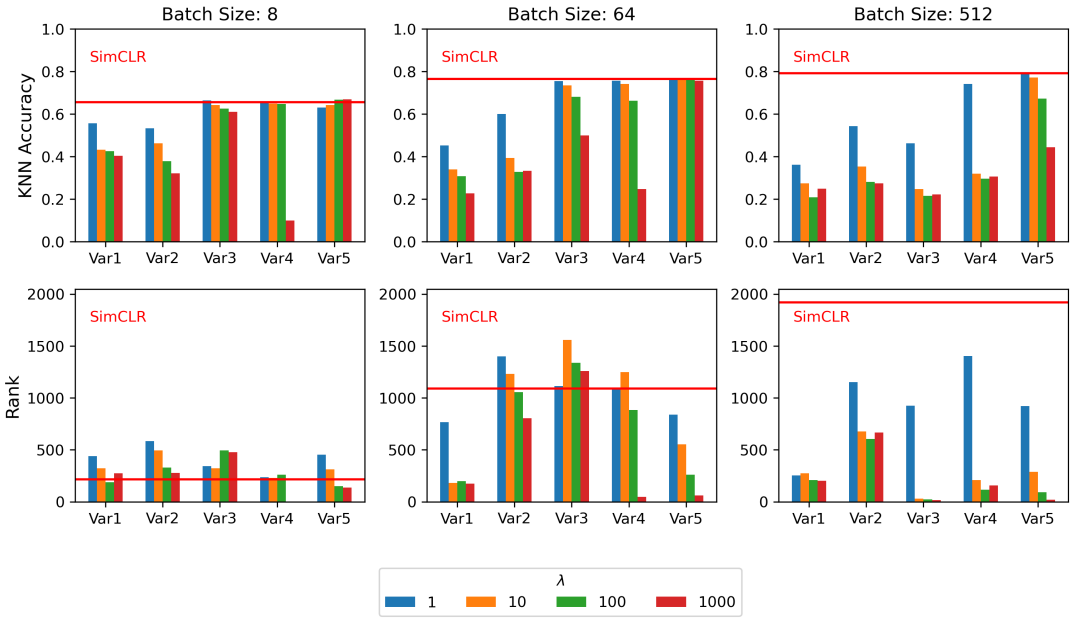

Figure 14: **Experiment 1:** The top row displays the accuracy achieved by a KNN classifier when applied to the generated representations for all five variants and different batch sizes, which vary by column. The bottom row illustrates the numerical rank of these representations.

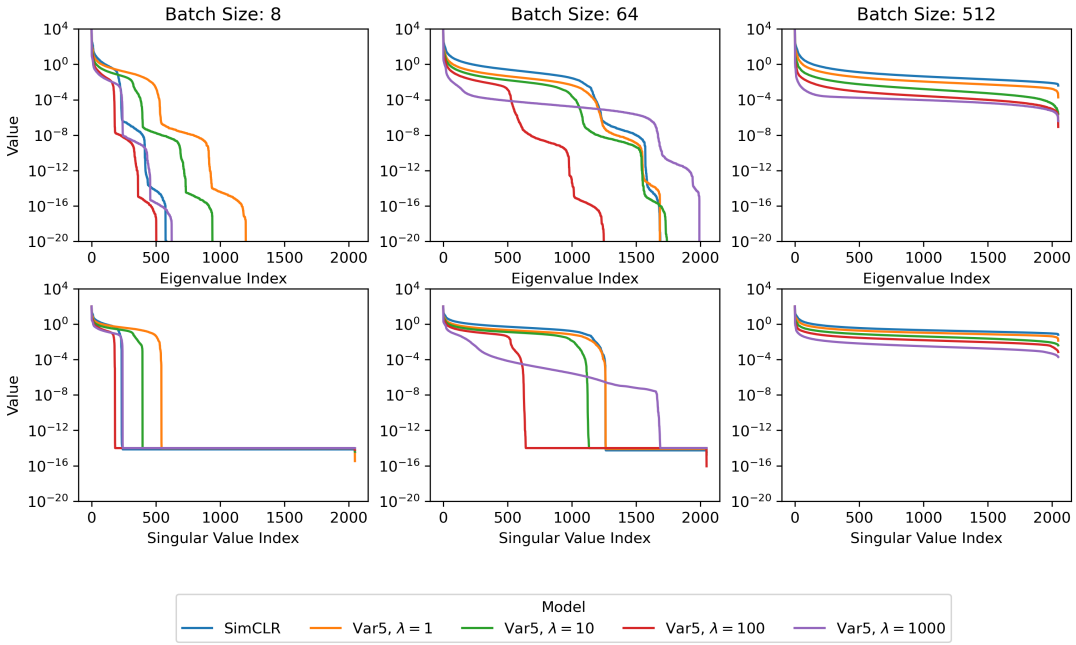

Figure 15: **Experiment 1:** Eigenvalue spectra of the covariance matrix of produced representations by the baseline model and the Variant 5 with different values for λ (top column) and the respective singular value spectra (bottom column) for different batch sizes varying by column.

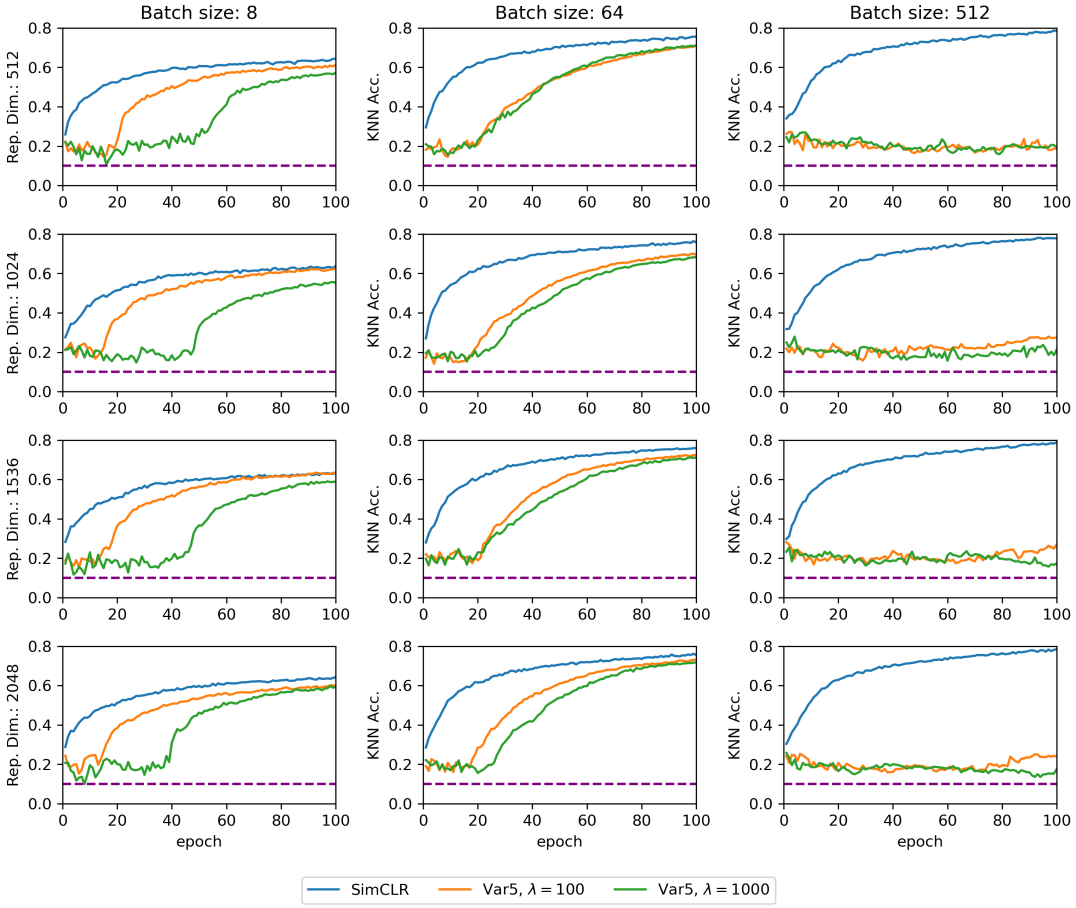

Figure 16: **Experiment 2:** Illustration of the progression of a KNN classifiers accuracy over each training epoch, applied to the representations of test data for different training batch sizes and different representation dimensionalities.The purple line marks the accuracy for random guessing (10%).

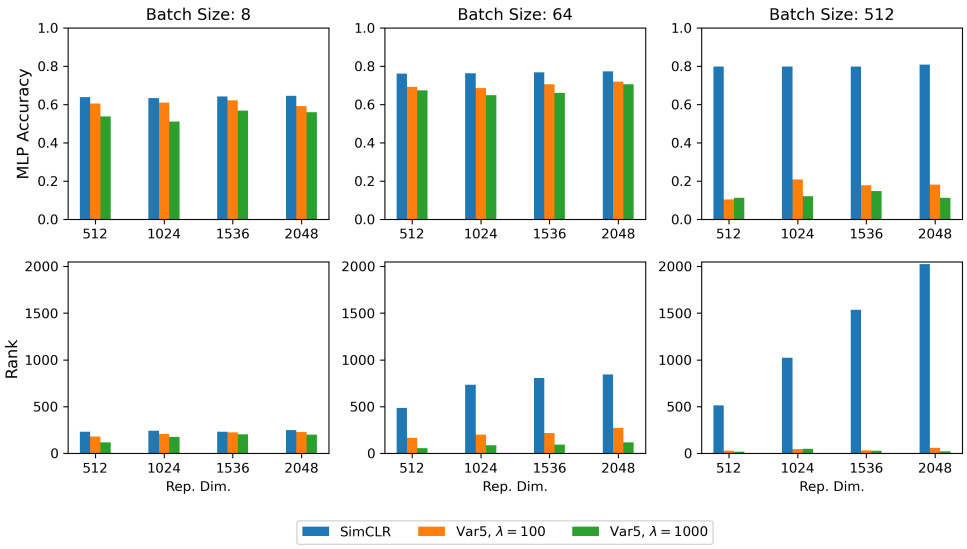

Figure 17: **Experiment 2:** The top row displays the accuracy achieved by a MLP classifier when applied to the generated representations for all four representation dimensionalities and different batch sizes, which vary by column. The bottom row illustrates the numerical rank of these representations.

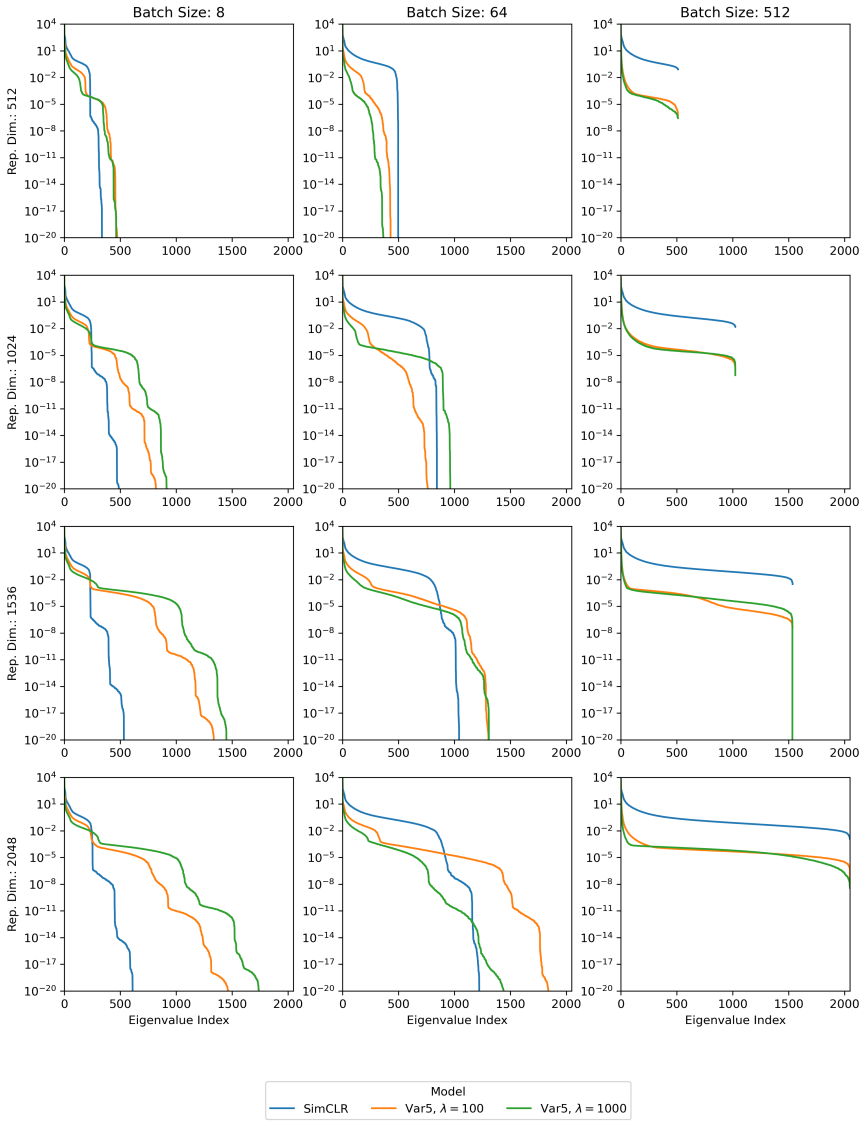

Figure 18: **Experiment 2:** Eigenvalue spectra of the covariance matrix of the generated representations by the base model and variant 5 with different values for λ for different embedding dimensionalities varying by row and different batch sizes varying by column.

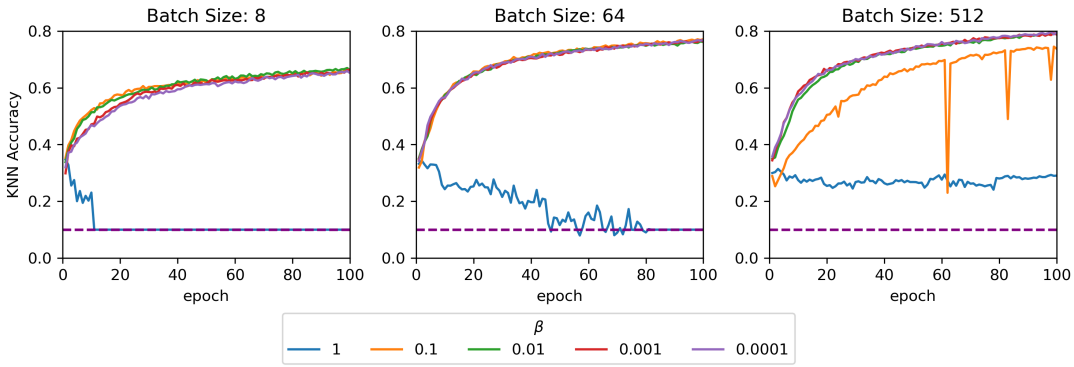

Figure 19: **Experiment 3:** Illustration of the progression of a KNN classifiers accuracy over each training epoch, applied to the representations of test data for different training batch sizes and different values for β. The purple line marks the accuracy for random guessing (10%).

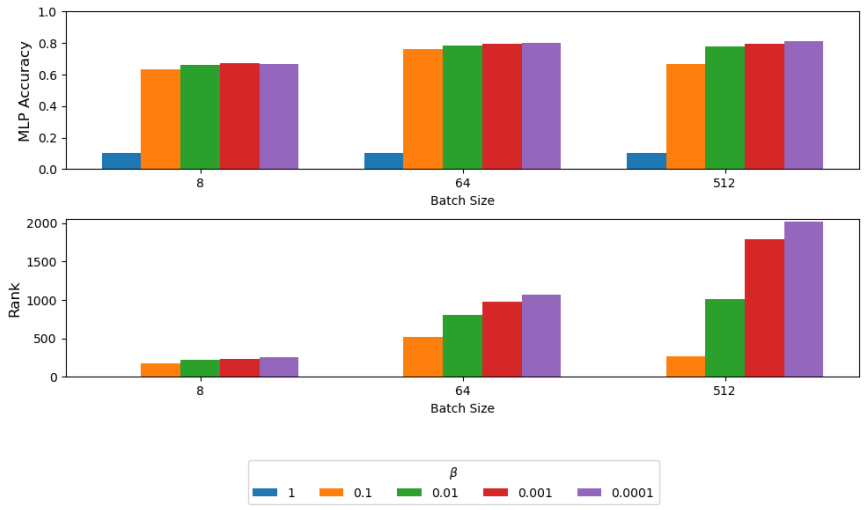

Figure 20: **Experiment 3:** The top row displays the accuracy achieved by a MLP classifier when applied to the generated representations for all five betas and different batch sizes, which vary by column. The bottom row illustrates the numerical rank of these representations.

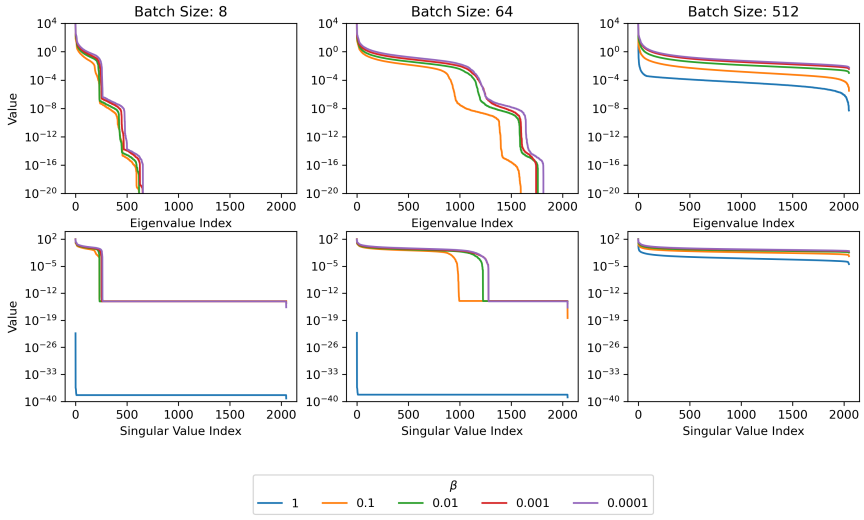

Figure 21: **Experiment 3:** Eigenvalue spectra of the covariance matrix of produced representations by the Variant 5 with different values for β (top column) and the respective singular value spectra (bottom column) for different batch sizes varying by column.

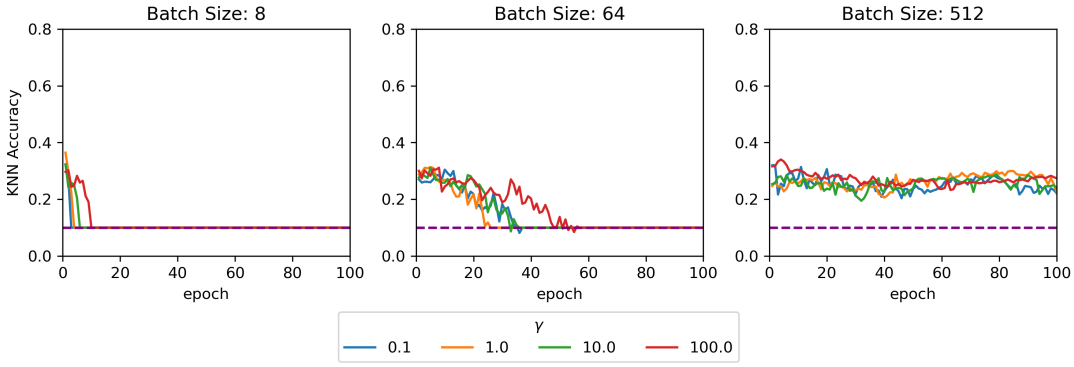

Figure 22: **Experiment 4:** Illustration of the progression of a KNN classifiers accuracy over each training epoch, applied to the representations of test data for different training batch sizes and different values for γ. The purple line marks the accuracy for random guessing (10%).

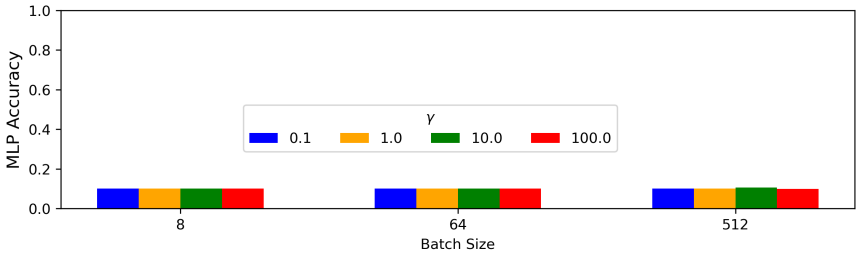

Figure 23: **Experiment 4:** The plot displays the accuracy achieved by a MLP classifier when applied to the generated representations for all four γ values and different batch sizes, which vary by column. Since the numerical rank of all representations was 1 the plot of the ranks was omitted for this experiment.

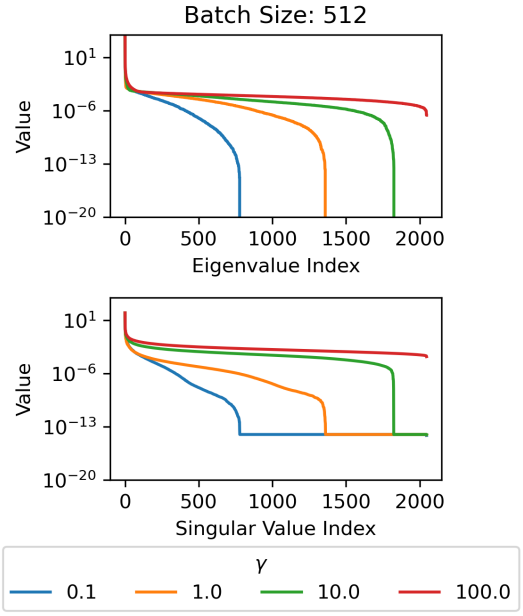

Figure 24: **Experiment 4:** Eigenvalue spectra of the covariance matrix of produced representations with different values for γ (top column) and the respective singular value spectra (bottom column) only for batch size 512 since the spectra for batch sizes 8 and 64 collapsed at index 1 and thus did not yield any more insights.

6. Discussion

Findings from Experiment 1 This experiment explored five different ways to apply the Topological Signature Loss within the SimCLR architecture, each variant focusing on different aspects of the data and representation layers. The goal was to see how these variants affect the embedding space and to assess their impact on downstream tasks.

In this exhaustive investigation it was discovered that versions V1 through V4 were ineffective due to poor performance.

Specifically, for V1 and V2, comparing the topological structures of vectorized image batches to their corresponding embeddings yields limited insights. Notably, minor image augmentations lead to significant changes in their vectorizations, while the embeddings, produced by functions f and g, should show less drastic alterations due to augmentation invariance by information distillation. This observation suggests that topological regularization might be detrimental in these two scenarios.

For V3, employing the Topological Signature Loss facilitated a topology-preserving dimension reduction by the projection head, similar to the approach used in the topology-preserving autoencoder by Moore et al. [1]. Despite this, such reduction did not enhance the performance in the subsequent task that utilizes the latent representation before the projection.

Versions V4 and V5 function similarly to the NT-Xent loss in a contrastive context. However, V4's performance continually declines as the value of λ increases, presenting no additional noteworthy effects.

Contrastingly, V5 demonstrates effective performance with both batch sizes 8 and 64. As shown in Figure 15, V5 maintains baseline performance while using fewer significant principal components.

A possible hypothesis concluded from this observations would be, that this efficiency in embedding relevant information for downstream tasks suggests that the Topological Signature Loss is beneficial in this setting. Topological regularization might be advantageous for SimCLR with smaller batch sizes. The rationale is that the topology of smaller batches, particularly with the strong augmentations used in CL, varies more significantly. While this variance tends to average out with larger batches, for smaller batch sizes, topological regularization may be crucial in offsetting this effect, potentially providing a more structured and beneficial representation space for downstream tasks.

Findings from Experiment 2 The focus here was on how the dimensionality of the representation space affects the model's ability to embed relevant information for downstream tasks. Different embedding dimensionalities are tested, both with and without

the implementation of Variant 5 of the Topological Signature Loss, to assess its influence on the quality of representations.

The hypothesis from Experiment 1, suggesting that V5 creates more efficient embeddings, was not supported, and in fact, contradicted by this experiment. The introduction of an additional layer for dimension reduction following function f did not significantly diminish performance of the baseline nor the versions using the Topological Signature Loss. However, the integration of the Topological Signature Loss was found to be ineffective in this context, and it even resulted in a substantial decline in performance, particularly for a batch size of 512.

Conclusively the topological regularisation as applied in this experiment does not lead to better aligned embeddings.

Findings from Experiment 3 This experiment investigated the interplay between the contrastive loss and the Topological Signature Loss (Variant 5) in the SimCLR framework. A new parameter, β, was introduced to adjust the balance between these two loss functions, aiming to understand their combined effect on the performance in downstream tasks.

A slight topological regularization appeared to have just a small negative effect on the performance in the downstream task when using smaller batch sizes. Both the eigenvalue and singular value spectra indicate a more rapid dimensional collapse when topological regularization is increased.

Furthermore, in this CL framework, relying solely on the Topological Signature Loss did not yield effective representations for subsequent tasks.

Findings from Experiment 4 In this experiment, the regularization component of the NT-Xent loss was excluded to evaluate the role of the Topological Signature Loss in regularization within the CL framework. The focus was on understanding the effectiveness of the Topological Signature Loss in maintaining data clustering relevant to semantic labels when the regularization aspect of NT-Xent is removed.

Only relying on the Topological Signature Loss for regularization did not yield useful representations at all, at least with the γ values used in this experiment. Two interesting trends emerged which suggest to further investigate this configuration with higher γ values, corresponding to a higher degree of topological regularization:

- Higher γ values led to a delayed decline of the KNNs performance in the training procedure for batch sizes 8 and 64.
- Higher γ values led to delayed dimensional collapse of the eigenvalue and singular value spectra for batch size 512. This stands in direct contrast to experiment 3 where the additional regularization through the Topological Signature Loss accelerated the dimensional collapse.

Criticism of the study of eigenvalue spectra The analysis of eigenvalue spectra conducted in the study often shows a discrepancy between the performance of the models in downstream tasks and their numerical rank or the amplitude of the eigenvalue spectrum. The interpretation of eigenvalue spectra falls into the realm of linear algebra (emphasizing linear relationships). Assessing the usefulness of a representation space solely through the analysis of the eigenvalue spectrum inherently presumes that the features or principal components are non-linearly disentangled. However, this aspect is often not true, so that the practical relevance can be low and the findings based on this assumption may be only of limited importance.

Problem with large batch sizes and the Topological Signature Loss In experiments 1-3, it became clear that topological regularization had a negative effect on downstream task performance when training was performed with a large batch size.

Unfortunately, the requirements with regard to the training of SimCLR and the application of the Topological Signature Loss are in direct conflict in terms of batch size: Moore et al. found in their paper on the topology-preserving autoencoder that smaller batch sizes are particularly suitable for its application. In particular they used a batch size of 82 for their experiments with CIFAR-10.

Chen et al. on the other hand, recognized that SimCLR could be trained particularly well with very large batch sizes (≥ 512) as the regularization part of the NT-Xent loss benefits from a high number of negative pairs.

In addition to SimCLR, however, there are also CL models such as MoCov [7] and VICReg [27], which achieve similarly good results decoupled from the batch size. For further investigations of the combination of CL and Topological Signature Losses in subsequent work, one of these models should be used.

7. Conclusion

The experiments conducted to investigate the impact of topological regularization within the SimCLR framework in CL yielded several important insights and directions for future research. These are summarized as follows:

Experiment 1: Variants of Topological Signature Loss Explored five different variants of applying the topological signature loss within the SimCLR architecture.

- **Ineffectiveness of V1-V4:** Variants 1 to 4 demonstrated suboptimal performance, indicating that their modes of integrating topological regularization do not contribute positively to the learning process or downstream task efficacy.
- **Effectiveness of V5:** Variant 5 emerged as a notable exception, maintaining relatively high performance, particularly for smaller batch sizes. This suggests that the manner of topological regularization in V5 might be beneficial in certain contexts, especially for smaller batches under strong augmentations.
- **Role of Batch Size and degree of topological Regularization:** Larger batch sizes typically correlated with improved performance in the downstream task, but this was not consistently observed across all variants. A aigher degree of regularization generally led to a decreased performance, with Variant 5 again being an exception in certain conditions.

Experiment 2: Impact of Representation Dimensionality Assessed the impact of representation dimensionality on the model's performance, especially when using Variant 5 of the topological signature loss and employed different embedding dimensionalities to explore the model's efficiency in embedding relevant information for downstream tasks.

- **Dimensionality and Performance:** Adjusting the number of dimensions in the representation space did not significantly affect model performance, implying that relevant information for classification might reside on a lower-dimensional manifold within the representation space.
- **Negative Impact of Topological Regularization:** An increased level of topological regularization generally led to a decreased performance, particularly noticeable for larger batch sizes.

Experiment 3: Balance between Loss Functions Introduced a new parameter, β, to explore the balance between the topological signature loss and the contrastive loss (NT-Xent loss).

- **Insufficiency of Topological Loss Alone:** Sole reliance on topological loss for smaller batch sizes significantly impaired the performance, indicating its insufficiency as the sole regularizer in this context.
- **Consistency Across Batch Sizes:** For beta values less than 1, the learning curves were similar across smaller batch sizes, suggesting consistency in the model's learned representations.

Experiment 4: Excluding NT-Xent Regularization Evaluated the effectiveness of the topological signature loss (Variant 5) for regularization by excluding the regularization term of the NT-Xent loss in the SimCLR framework.

- **Need for Regularization:** Eliminating the regularization component of the NT-Xent loss and relying solely on topological regularization did not yield performance improvements, at least with the degree of regularization used in this configuration, highlighting the importance of a balanced approach between invariance and regularization in CL.

General Observations and Criticism

- **Eigenvalue Spectra Analysis:** The analysis often revealed a disconnection between the models' downstream task performance and their numerical rank or eigenvalue spectra, suggesting limitations in using the eigenvalue spectrum analysis as a quality indicator.
- **Large Batch Size Problem:** Topological regularization consistently showed a negative effect on performance with large batch sizes, posing a challenge for models like SimCLR which benefit from larger batch sizes.

Future Research Directions

- **Continuing Experiment 4:** Higher γ values corresponding to a higher degree of topological regularization may lead to better results.
- **Exploring Other CL Models:** Future research should consider other CL models, such as MoCo [6] or VICReg [27], which are less dependent on large batch sizes.
- **Exploring Other Loss Functions:** Besides the topological signature loss [1], there exist other loss functions built on PH like the connectivity loss (Definition 31), which could be of use within the CL framework.

In conclusion, while topological regularization, particularly in the form applied in Variant 5, shows potential in enhancing CL models, its integration is complex and necessitates careful consideration of model architecture and training dynamics. Overall these findings provide a foundation for future research in embedding topological aspects into CL frameworks.

List of Figures

1. The same underlying data presented in cartesian and polar coordinates. 10
2. The general contrastive learning framework consisting of an input sample $x \in X$ two different augmentations $t_i, t'_i \in T$, the augmented versions of the image $x_i, x'_i \in \mathbb{R}^N$ the encoder network $f : \mathbb{R}^N \to \mathbb{R}^M$ the representations $y_i, y'_i \in \mathbb{R}^M$ the projection head $g : \mathbb{R}^M \to \mathbb{R}^K$ and the embeddings $z_i, z'_i \in \mathbb{R}^K$ on which the contrastive loss is then calculated. 14
3. Different augmentations applied to an image. 20
4. A 0-, a 1-, a 2- and a 3-simplex. 22
5. A 3-simplex and its 2-dimensional faces. 23
6. A 3-simplicial complex on the left and a 2-subcomplex on the right. . 23
7. A 3-simplicial complex on the left its 1-skeleton on the right. 24
8. A 2-simplex. 25
9. An example persistence diagram displaying the time of birth and time of death of four homology classes. 29
10. An overview of the method presented by Moore et al. in [1]. The graphic was taken from the original paper. 31
11. Variants of topological signature loss calculation in Experiment 1 . . . 34
12. **Experiment 1:** Illustration of the dynamic balance between the topological signature loss and the contrastive loss during the training. Each curve represents a different configuration, showing how varying the hyperparameter λ influences the ratio of these two loss functions for a different different batch size in each column. 40
13. **Experiment 1:** Illustration of the progression of a KNN classifiers accuracy over each training epoch, applied to the representations of test data for different training batch sizes and all variants of the loss calculation with different values for λ, as well as the baseline model. The purple line marks the accuracy for random guessing (10%). 44
14. **Experiment 1:** The top row displays the accuracy achieved by a KNN classifier when applied to the generated representations for all five variants and different batch sizes, which vary by column. The bottom row illustrates the numerical rank of these representations. 45
15. **Experiment 1:** Eigenvalue spectra of the covariance matrix of produced representations by the baseline model and the Variant 5 with different values for λ (top column) and the respective singular value spectra (bottom column) for different batch sizes varying by column. 46
16. **Experiment 2:** Illustration of the progression of a KNN classifiers accuracy over each training epoch, applied to the representations of test data for different training batch sizes and different representation dimensionalities.The purple line marks the accuracy for random guessing (10%). 47

17. **Experiment 2:** The top row displays the accuracy achieved by a MLP classifier when applied to the generated representations for all four representation dimensionalities and different batch sizes, which vary by column. The bottom row illustrates the numerical rank of these representations. 48
18. **Experiment 2:** Eigenvalue spectra of the covariance matrix of the generated representations by the base model and variant 5 with different values for λ for different embedding dimensionalities varying by row and different batch sizes varying by column. 49
19. **Experiment 3:** Illustration of the progression of a KNN classifiers accuracy over each training epoch, applied to the representations of test data for different training batch sizes and different values for β. The purple line marks the accuracy for random guessing (10%). 50
20. **Experiment 3:** The top row displays the accuracy achieved by a MLP classifier when applied to the generated representations for all five betas and different batch sizes, which vary by column. The bottom row illustrates the numerical rank of these representations. 50
21. **Experiment 3:** Eigenvalue spectra of the covariance matrix of produced representations by the Variant 5 with different values for β (top column) and the respective singular value spectra (bottom column) for different batch sizes varying by column. 51
22. **Experiment 4:** Illustration of the progression of a KNN classifiers accuracy over each training epoch, applied to the representations of test data for different training batch sizes and different values for γ. The purple line marks the accuracy for random guessing (10%). 51
23. **Experiment 4:** The plot displays the accuracy achieved by a MLP classifier when applied to the generated representations for all four γ values and different batch sizes, which vary by column. Since the numerical rank of all representations was 1 the plot of the ranks was omitted for this experiment. 52
24. **Experiment 4:** Eigenvalue spectra of the covariance matrix of produced representations with different values for γ (top column) and the respective singular value spectra (bottom column) only for batch size 512 since the spectra for batch sizes 8 and 64 collapsed at index 1 and thus did not yield any more insights. 52

Glossary

β-VAE β - Variational Auto Encoder

AE Autoencoder

CL Contrastive Representation Learning

CNN Convolutional Neural Network

DL Deep Learning

KNN K-Nearest Neighbor

LSTM Long Short Term Memory

ML Machine Learning

MLP Multi-layer Perceptron

OCC One Class Classification

PCA Principal Component Analysis

PH Persistent Homology

RL Representation Learning

TDA Topological Data Analysis

VAE Variational Auto Encoder

List of Tables

1. The groups of the simplex in Figure 8. 25
2. Hyperparameters for model training in Experiment 1 35
3. Hyperparameters for model training in Experiment 2 36
4. Hyperparameters for model training in Experiment 3 36
5. Hyperparameters for model training in Experiment 4 37
6. List of used python packages 65

References

[1] Michael Moor, Max Horn, Bastian Rieck, and Karsten Borgwardt. "Topological Autoencoders". In: *International Conference on Machine Learning* 37 (2019), pp. 7001–7011. DOI: 10.48550/arxiv.1906.00722 (cit. on pp. 7, 21, 30, 31, 33, 34, 53, 57, 65).

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 9, 11).

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (Aug. 2013), pp. 1798–1828. DOI: 10.1109/TPAMI.2013.50 (cit. on pp. 9, 12, 19).

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Everest Hinton. "A Simple Framework for Contrastive Learning of Visual Representations". In: 2020 (cit. on pp. 16–19, 33, 39).

[5] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. "Unsupervised Feature Learning via Non-parametric Instance Discrimination". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3733–3742. DOI: 10.1109/CVPR.2018.00393 (cit. on p. 16).

[6] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. "Momentum Contrast for Unsupervised Visual Representation Learning". In: June 2020, pp. 9726–9735. DOI: 10.1109/CVPR42600.2020.00975 (cit. on pp. 16, 17, 57).

[7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. *Improved Baselines with Momentum Contrastive Learning*. 2020. arXiv: 2003.04297 [cs.CV] (cit. on pp. 17, 55).

[8] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments". 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. 9912–9924 (cit. on p. 17).

[9] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. "Improved baselines with momentum contrastive learning". In: *arXiv preprint arXiv:2003.04297* (2020) (cit. on p. 18).

[10] Kartik Gupta, Thalaiyasingam Ajanthan, Anton van den Hengel, and Stephen Gould. "Understanding and improving the role of projection head in self-supervised learning". In: *arXiv preprint arXiv:2212.11491* (2022) (cit. on pp. 18, 39).

[11] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. "Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination". In: *CoRR* abs/1805.01978 (2018). arXiv: 1805.01978 (cit. on p. 20).

[12] Gurjeet Singh, Facundo Memoli, Tigran Ishkhanov, Guillermo Sapiro, Gunnar Carlsson, and Dario L. Ringach. "Topological analysis of population activity in visual cortex". In: *Journal of Vision* 8 (2008), pp. 11–11. DOI: 10.1167/8.8.11 (cit. on p. 21).

[13] Bastian Rieck, Tristan Yates, Christian Bock, Karsten Borgwardt, Guy Wolf, Nicholas Turk-Browne, and Smita Krishnaswamy. "Uncovering the Topology of Time-Varying fMRI Data using Cubical Persistence". In: *Advances in neural information processing systems* 33 (2020), pp. 6900–6912 (cit. on p. 21).

[14] Kelin Xia and Guo-Wei Wei. "Multidimensional persistence in biomolecular data". In: *Journal of Computational Chemistry* 36 (2015), pp. 1502–1520. DOI: https://doi.org/10.1002/jcc.23953 (cit. on p. 21).

[15] Christoph Hofer, Roland Kwitt, Marc Niethammer, and Mandar Dixit. "Connectivity-Optimized Representation Learning via Persistent Homology". In: *International Conference on Machine Learning* 36 (2019), pp. 2751–2760 (cit. on pp. 21, 31).

[16] Chao Chen, Xiuyan Ni, Qinxun Bai, and Yusu Wang. "A Topological Regularizer for Classifiers via Persistent Homology". In: *Proceedings of Machine Learning Research* 89 (2019), pp. 2573–2582 (cit. on p. 21).

[17] Stefan Huber. "Persistent Homology in Data Science". In: *Data Science – Analytics and Applications* 3 (2021), pp. 81–88 (cit. on pp. 21, 23).

[18] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction.* American Mathematical Society, 2010, p. 241 (cit. on pp. 21, 22, 29).

[19] Herbert Edelsbrunner and John Harer. "Persistent homology - a survey". In: *Discrete and Computational Geometry - DCG* 453 (2008). DOI: 10.1090/conm/453/08802 (cit. on p. 21).

[20] Robert W. Ghrist. *Elementary applied topology.* Createspace Independent Publishing Platform, 2014 (cit. on p. 21).

[21] Gunnar Carlsson. "Topology and Data". In: *BULLETIN of the American Mathematical Society* 46 (2 2009), pp. 255–308 (cit. on p. 21).

[22] L. Vietoris. "Über den höheren Zusammenhang kompakter Räume und eine Klasse von zusammenhangstreuen Abbildungen". In: *Mathematische Annalen* 97 (1 1927). DOI: 10.1007/BF01447877 (cit. on p. 27).

[23] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. *CIFAR-10 (Canadian Institute for Advanced Research).* URL: http://www.cs.toronto.edu/~kriz/cifar.html (cit. on p. 33).

[24] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* Ed. by Yoshua Bengio and Yann LeCun. 2015 (cit. on pp. 35, 38).

[25] Ido Ben-Shaul, Ravid Shwartz-Ziv, Tomer Galanti, Shai Dekel, and Yann LeCun. "Reverse Engineering Self-Supervised Learning". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023 (cit. on p. 36).

[26] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. "Understanding Dimensional Collapse in Contrastive Self-supervised Learning". In: *CoRR* abs/2110.09348 (2021). arXiv: 2110.09348 (cit. on p. 39).

[27] Adrien Bardes, Jean Ponce, and Yann LeCun. "VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning". In: *CoRR* abs/2105.04906 (2021). arXiv: 2105.04906 (cit. on pp. 55, 57).

A. Used Software Packages

The pytorch implementation of SimCLR und the experiments of this work can be found under https://github.com/simonschindler/masterthesisSiS.

For the calculation of the Topological Signature Loss (Definition 29) the implementation of the authors of the original paper [1] was used, which can be found under https://github.com/BorgwardtLab/topological-autoencoders.

The following python packages were used:

Table 6: List of used python packages

Package	Version
absl-py	1.3.0
aiohttp	3.8.3
aiosignal	1.3.1
alembic	1.12.0
anyio	4.1.0
appdirs	1.4.4
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
astunparse	1.6.3
async-lru	2.0.4
async-timeout	4.0.2
attrs	22.2.0
Babel	2.13.1
backcall	0.2.0
backports.functools-lru-cache	1.6.4
beautifulsoup4	4.12.2
black	22.10.0
bleach	6.1.0
blinker	1.6.3
brotlipy	0.7.0
cached-property	1.5.2
cachetools	5.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	2.1.1
click	8.1.3
cloudpickle	2.2.1
colorama	0.4.6

Continued on next page

Table 6 – continued from previous page

Package	Version
comm	0.1.2
contourpy	1.0.6
cryptography	39.0.0
cycler	0.11.0
Cython	0.29.32
databricks-cli	0.18.0
debugpy	1.6.4
decorator	5.1.1
defusedxml	0.7.1
Deprecated	1.2.13
dill	0.3.6
dm-tree	0.1.7
docker	6.1.3
docopt	0.6.2
entrypoints	0.4
etils	0.9.0
exceptiongroup	1.2.0
executing	1.2.0
fastjsonschema	2.16.2
Flask	2.3.3
flatbuffers	23.1.4
fonttools	4.38.0
fqdn	1.5.1
frozenlist	1.3.3
gast	0.4.0
gitdb	4.0.11
GitPython	3.1.40
google-auth	2.15.0
google-auth-oauthlib	0.4.6
google-pasta	0.2.0
googleapis-common-protos	1.57.1
greenlet	3.0.0
grpcio	1.47.1
gunicorn	21.2.0
h5py	3.7.0
hopcroftkarp	1.2.5
idna	3.4
importlib-metadata	6.0.0
importlib-resources	5.10.2
ipykernel	6.19.4

Continued on next page

Table 6 – continued from previous page

Package	Version
ipympl	0.9.2
ipython	8.8.0
ipython-genutils	0.2.0
ipywidgets	8.0.4
isoduration	20.11.0
isort	5.12.0
itsdangerous	2.1.2
jedi	0.18.2
Jinja2	3.1.2
joblib	1.2.0
json5	0.9.14
jsonpickle	3.0.2
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter_client	7.4.8
jupyter_core	5.1.2
jupyter-events	0.9.0
jupyter-lsp	2.2.1
jupyter_server	2.11.1
jupyter_server_terminals	0.4.4
jupyterlab	4.0.9
jupyterlab_pygments	0.3.0
jupyterlab_server	2.25.2
jupyterlab-widgets	3.0.5
kaggle	1.5.16
keras	2.10.0
Keras-Preprocessing	1.1.2
kiwisolver	1.4.4
kmapper	2.0.1
llvmlite	0.40.1
Mako	1.2.4
Markdown	3.4.1
MarkupSafe	2.1.1
matplotlib	3.6.2
matplotlib-inline	0.1.6
mistune	3.0.2
mlflow	2.7.1
multidict	6.0.4
munch	2.5.0

Continued on next page

Table 6 – continued from previous page

Package	Version
munkres	1.1.4
mypy-extensions	0.4.3
nbclient	0.9.0
nbconvert	7.11.0
nbformat	5.7.1
nest-asyncio	1.5.6
notebook	7.0.6
notebook_shim	0.2.3
numba	0.57.1
numpy	1.24.1
oauthlib	3.2.2
opencv-python	4.8.1.78
opt-einsum	3.3.0
overrides	7.4.0
packaging	22.0
pandas	1.5.2
pandocfilters	1.5.0
parso	0.8.3
pathspec	0.10.3
persim	0.3.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.4.0
pip	22.3.1
pkgutil_resolve_name	1.3.10
platformdirs	2.6.2
plotly	5.18.0
ply	3.11
pooch	1.6.0
prometheus-client	0.19.0
promise	2.3
prompt-toolkit	3.0.36
protobuf	4.21.12
psutil	5.9.4
ptyprocess	0.7.0
pure-eval	0.2.2
py-cpuinfo	9.0.0
pyarrow	13.0.0
pyasn1	0.4.8
pyasn1-modules	0.2.7

Continued on next page

Table 6 – continued from previous page

Package	Version
pycparser	2.21
Pygments	2.14.0
PyJWT	2.6.0
pyOpenSSL	23.0.0
pyparsing	3.0.9
PyQt5	5.15.7
PyQt5-sip	12.11.0
pyrsistent	0.19.3
PySocks	1.7.1
python-dateutil	2.8.2
python-json-logger	2.0.7
python-slugify	8.0.1
pytz	2022.7
pyu2f	0.1.5
PyYAML	6.0.1
pyzmq	24.0.1
querystring-parser	1.2.4
referencing	0.31.0
requests	2.31.0
requests-oauthlib	1.3.1
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
ripser	0.6.4
rpds-py	0.13.1
rsa	4.9
sacred	0.8.4
scikit-learn	1.2.0
scikit-tda	1.0.0
scipy	1.10.0
Send2Trash	1.8.2
setuptools	65.6.3
sip	6.7.5
six	1.16.0
smmap	5.0.1
sniffio	1.3.0
soupsieve	2.5
SQLAlchemy	2.0.22
sqlparse	0.4.4
stack-data	0.6.2
tabulate	0.9.0

Continued on next page

Table 6 – continued from previous page

Package	Version
tadasets	0.0.4
tenacity	8.2.3
tensorboard	2.10.1
tensorboard-data-server	0.6.1
tensorboard-plugin-wit	1.8.1
tensorflow	2.10.0
tensorflow-datasets	4.8.1+nightly
tensorflow-estimator	2.10.0
tensorflow-metadata	1.12.0
termcolor	2.2.0
terminado	0.18.0
text-unidecode	1.3
threadpoolctl	3.1.0
tinycss2	1.2.1
tokenize-rt	5.0.0
toml	0.10.2
tomli	2.0.1
torch	1.13.0.post200
torchvision	0.14.0a0+2ba5a5d
tornado	6.2
tqdm	4.64.1
traitlets	5.8.0
types-python-dateutil	2.8.19.14
typing_extensions	4.4.0
umap-learn	0.3.10
unicodedata2	15.0.0
uri-template	1.3.0
urllib3	1.26.13
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4
Werkzeug	3.0.1
wheel	0.38.4
widgetsnbextension	4.0.5
wrapt	1.14.1
yarl	1.8.2
zipp	3.11.0