Bayesian Contrastive Learning on Topological Structures

Alexander Möllers
5184746
EEMCS, TU Delft

Supervisors:
Prof. Elvin Isufi
Prof. Hanne Kekkonen
Dr. Vincent Fortuin
Alexander Immer

Abstract

In this thesis we develop a Bayesian approach to graph contrastive learning and propose a new uncertainty measure based on the disagreement in likelihood due to different positive samples. Moreover, we extend contrastive learning to simplicial complexes and show that it can be used to generate high-quality representations of edge flow data.

A thesis to obtain the degree of Master of Science in Applied Mathematics with a specialisation in Stochastics at the faculty of Electrical Engineering, Mathematics and Computer Science (EECMS), TU Delft. To be defended on the 8th of November 2023.

Preface

The truth is that life is not always easy and sometimes it takes unexpected turns. In some sense my journey is a testament to that unpredictability. During high school my talents seemed to lie in the realm of languages, then I started a bachelors in industrial engineering to become a manager, and ended up with a master's degree in mathematics. All this happened across continents and cultures and I had the opportunity to live in places as diverse as China, the Netherlands, Switzerland, and Argentina. Each transition was fraught with its own challenges and often came with a shift in hobbies and interests—from dreams of becoming a soccer player to speaking Chinese, from climbing to electronic music and meditation. All this variability and unexpectedness made me realize the importance of defining how I want to live, what matters to me, and which principles guide me in times when the correct path is not clear to see.

The answer I found for myself in the past years is to work on my relationships with the people I love and to reduce the influence societal norms and expectations have on my life. I have learned to openly express my feelings, to prioritize healthy relationships over minor squabbles, and that small gestures go a long way. More than anything this thesis and the resulting papers are dedicated to my loving family Gaby, Christian and Jani and to my best friends Alex, Andrés, Johann, Sophie, Marco, Mariano, Lena, Lena, Rocío and Vanessa who make this life so worthwhile and beautiful. I am the most fortunate person to have you by my side and am looking forward to the years to come. Thank you for everything.

On the professional side, I would like to thank Vincent Fortuin, Alexander Immer and Elvin Isufi for guiding me throughout my master thesis. They have been an inspiration. Vincent taught me to broaden my perspective, to see the bigger picture in research and to understand how many incremental works can combine to an important contribution. From Alex I learned about the importance of being an empiricist and inherited the desire to conduct honest, well founded, research that helps the field and the community to progress. Last but not least, Elvin taught me to focus on the underlying idea in a paper and to critically interpret the numbers in a world where everybody always beats the baseline.

I would also like to give thanks to Oscar Portoles Marin who supervised me during my bachelor studies. His unwavering trust and belief in my capabilities have had a lasting impact on my professional life up until this day. Moreover, I would like to thank the members of my thesis committee, Hanne Kekkonen and Leo van Iersel, for taking the time to review this work and help me graduate.

Alexander Julien Möllers, on the 31st of October 2023, in Göttingen (Germany)

Mathematical Notation and Abbreviations

Throughout the thesis we use lower case letters for scalars, caligraphic letters for sets, bold lower case letters for vectors and bold upper case letters for matrices. We have the following abbreviations and symbols:

Symbol	Explanation
$\mathbb{R}, \mathbb{R}_+$	set of real and positive real numbers
$\mathbf{x}$	data point, typically a vector containing the different node or edge values
$\mathbf{x}'$	augmentation or contrast of a data point.
$\mathcal{D}$	set containing the observed data. Can contain data points, labels etc.
$\mathbf{z}$	embedding vector obtained by passing $\mathbf{x}$ through an encoder
$\mathbf{z}'$	embedding vector obtained by passing $\mathbf{x}'$ through an encoder
$g_{\boldsymbol{w}}(\mathbf{x})$	function mapping $\mathbf{x}$ to some output parameterized by $\boldsymbol{w}$, usually a neural network
$f(\mathbf{z}, \mathbf{z}')$	similarity function for two embedding. Usually the normalised cosine similarity
$p(\cdot)$	probability density of the argument
$q_{\boldsymbol{\theta}}(\cdot)$	parameterized variational distribution used for approximation
$\mathcal{L}(\cdot)$	loss or cost function that is being optimized
$f \circ g$	composition of two functions f and g

Contents

1 Introduction

Research on neural networks has made significant progress in improving their performance on classification tasks. Nevertheless, they still perform poorly in scarce data settings and do not use the information contained in unlabeled data (Alzubaidi et al., 2021). To address this, researchers developed so-called contrastive learning (CL) methods that alleviate the two issues (Jaiswal et al., 2021). These methods use positive and negative examples for each data point and train a neural network (the encoder) such that it maps the positives close to each other and the negatives further apart in an embedding space (e.g. Chen et al., 2020, He et al., 2020, Zhu et al., 2020, You et al., 2020). In image processing, a positive example for a picture of a German Shepherd could be a Golden Retriever as they stem from the same underlying latent class dog (Figure 1). A cat would be a negative example. In most applications, data augmentations such as colour jittering or cropping of the original data point (anchor) are used to generate the positive examples and the negative ones are random samples from the data. Problem-specific knowledge can be imbued into the learner via the augmentation design and the generated embeddings containing that knowledge can then be fed into a downstream classifier. To optimize the parameters of the encoder contrastive losses are used, among which the InfoNCE objective (van den Oord et al., 2018) has been applied to train many of the best-performing models.

While the contrastive learning approach has been successful in a wide variety of domains, ranging from graph classification to computer vision, understanding and extending it is still a flourishing field of research (e.g. You et al., 2021, Bardes et al., 2022, Zbontar et al., 2021). Two areas that have been largely unexplored are Bayesian interpretations of contrastive learning and the generalization of the developed methods to topological structures beyond graphs. Therefore, this thesis researches the InfoNCE objective with respect to these two points. In particular, we investigate (1) how to implement Bayesian InfoNCE learning and how to use it to measure uncertainties and (2) how InfoNCE learning can be extended to a class of topological objects called simplicial complexes (SCs). A Bayesian approach could facilitate the adoption of CL methods in high-risk applications such as the medical field where accurate uncertainty quantification is indispensable. Furthermore, SCs have been shown to be particularly useful for modelling flows (e.g., mass, energy, information, or trajectories) and incorporating them into a contrastive learning framework promises to yield a method that can generate good representations for these kinds of data (Barbarossa and Sardellitti, 2020, Roddenberry et al., 2022).

To enable the reader to read about each of these topics independently from each other, the

thesis is divided into these two parts. While the methods developed in the first part are of a general nature, we apply them to graphs and show that they can be used to achieve improvements on existing baseline tasks in that domain. This results in the following contributions that we make in the first part:

(i) We propose variational graph constrastive learning (VGCL), a Bayesian approach to contrastive learning that incorporates the epistemic uncertainty in the weights in a principled manner. It defines Gaussian priors over the network parameters and learns the related posterior distributions. We further show that the approximating variational family can be regularized to improve downstream accuracy for node classification tasks.

(ii) We propose a new approach for measuring uncertainty in contrastive learning based on the disagreement between positive samples and call it the Contrastive Model Disagreement Score (CMDS). In contrast to existing methods, it can incorporate Bayesian uncertainty and is directly related to the likelihood of a probabilistic model of contrastive learning. On graphs, it can be used to outperform currently existing uncertainty measures.

The contributions we make in the second part are:

(i) We propose Contrastive Learning on Simplicial Complexes (SCL), design related augmentations and show that it can be used to generated effective representation for edge flow data.

(ii) We design augmentation methods that are able to incorporate knowledge about the spectral structure of simplicial complexes. This can be used to introduce information related to the so-called Hodge decomposition of the spectrum into the embeddings. The latter is foundational in numerous applications.

(iii) We introduce a reweighing of the negative examples based on the similarity of their Hodge components to encourage a spectrally organized embedding space.

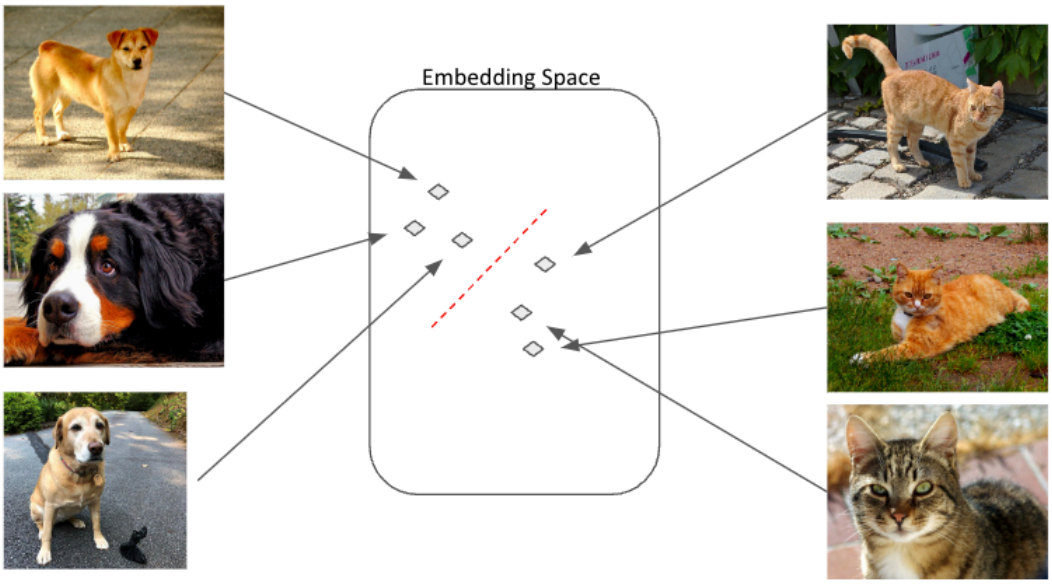

Figure 1: A conceptual example of contrastive learning with animal images. Dogs are similar to each other and mapped to the same place in the embedding space. Cats are mapped to a place further away. The resulting embeddings reside in the euclidean space and are often separable by a linear classifier (red line). They can for example be used in a subsequent classification task.

1.1 Structure of the Thesis

We divide this thesis into two parts (1) Variational Graph Contrastive Learning and (2) Contrastive Learning on Simplicial Complexes. Each part contains a background section, a methodology, experiments, results and related discussions. To read the second part knowledge of contrastive learning and neural networks is required (Sections 2.1 and 2.3).

Part I

Variational Graph Contrastive Learning

2 Background I

2.1 Neural Networks and their Bayesian Interpretation

2.1.1 Neural Networks

Many problems in machine learning boil down to approximating some ideal function g^* (Goodfellow et al., 2016). Usually, this is done by optimizing the parameters $\boldsymbol{w}$ of a surrogate mapping $g_{\boldsymbol{w}}(\mathbf{x})$. For example, in a linear regression, the true relationship between the data $\mathbf{x} \in \mathbb{R}^F$ and the dependent variable $\mathbf{y} \in \mathbb{R}^M$ is approximated with a linear function of the form $\mathbf{y} = \mathbf{W}\,\mathbf{x} + \mathbf{b}$, $\mathbf{b} \in \mathbb{R}^M$ is the offset or bias, $\mathbf{W} \in \mathbb{R}^{M \times F}$ is the weight matrix, and $\boldsymbol{w} = (\mathbf{W}, \mathbf{b})$.

Often the ideal function g^* is complex and nonlinear. Thus, to learn it accurately, the function $g_{\boldsymbol{w}}(\mathbf{x})$ needs to be able to represent a sufficiently large function space. A strategy to achieve this, is to take $g_{\boldsymbol{w}}(\mathbf{x})$ to be a composition of K functions of the form $g_{\boldsymbol{w}^{[i]}}(\mathbf{x}) = \sigma\left(\mathbf{W}^{[i]}\,\mathbf{x} + \mathbf{b}^{[i]}\right)$ with σ being a nonlinear transformation. Hereby, the dimensions of the in and outputs for each function can be freely chosen and the dimensions of the weights $\mathbf{W}^{[i]}$ and biases $\mathbf{b}^{[i]}$ are adjusted accordingly. This can be written mathematically as:

$$g_{\boldsymbol{w}}(\mathbf{x}) = \mathcal{F}(g_{\boldsymbol{w}^{[K]}} \circ \ldots \circ g_{\boldsymbol{w}^{[2]}} \circ g_{\boldsymbol{w}^{[1]}}(\boldsymbol{x})) \tag{1}$$

where $\mathcal{F}(x)$ is some function that transforms the output of the neural network into the format that is required for the problem at hand and $\boldsymbol{w} = (\boldsymbol{w}^{[1]}, ..., \boldsymbol{w}^{[K]})$. This construction of $g_{\boldsymbol{w}}(\boldsymbol{x})$ is commonly known as a (deep) neural network with K (hidden) layers. If the weights are visualised as connections between the different components of the data, it leads to a network-like structure as displayed in Figure 2. The so-called activation function $\sigma(\mathbf{x})$ is often taken to be the rectified linear unit (ReLU), the Sigmoid or the hyperbolic tangent (Tanh).

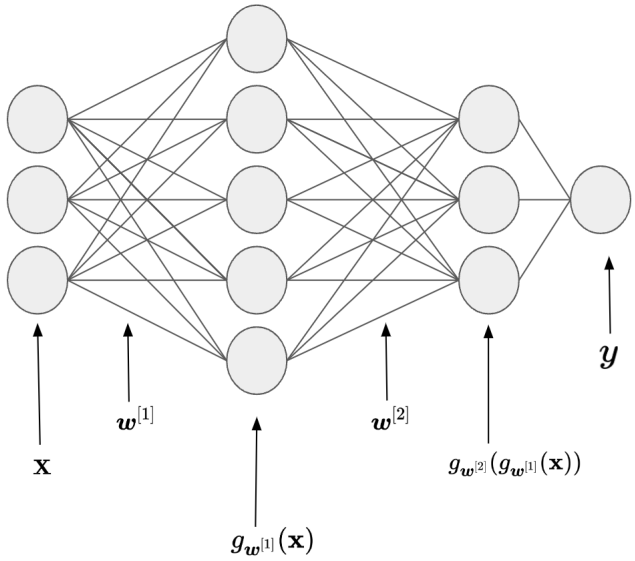

Figure 2: A two layered neural network which takes some input data $\mathbf{x} \in \mathbb{R}^3$ with three features. The grey circles represent computational units (neurons), that arise from computing a weighted sum of the input coupled with a nonlinear transformation. In this example the output is one-dimensional and we model it as $y = \mathcal{F}(g_{\boldsymbol{w}^{[2]}}(g_{\boldsymbol{w}^{[1]}}(\mathbf{x})))$ with $g_{\boldsymbol{w}^{[1]}}(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}^4$, $g_{\boldsymbol{w}^{[2]}}(\mathbf{x}) : \mathbb{R}^4 \to \mathbb{R}^3$ and $\mathcal{F} : \mathbb{R}^3 \to \mathbb{R}^1$. The latter is the sum of the values in the final neural network layer, i.e. $\mathcal{F}(\mathbf{x}) = \mathbf{W}^{[3]}\,\mathbf{x} + \mathbf{b}^{[3]}$. Such an output is for instance used in regression problems where the task is to predict a single number.

A theoretical result that supports using neural networks to approximate g^* is the Universal Approximation Theorem (Hornik et al., 1989). It states that a neural network with a single hidden layer and a sufficiently large number of neurons, coupled with certain mild assumptions about the nonlinearity σ, can approximate any continuous function defined on a compact subsets of the real coordinate space. Nevertheless, in practice the number of required neurons to approximate a complex function is often very large and it is more computationally efficient to stack them in a layer-wise fashion, which gives rise to the concept of deep neural networks. Universal approximation results for the arbitrary depth case and for structures such as graphs exist (Lu et al., 2017, Brüel Gabrielsson, 2020).

2.1.2 Optimizing Neural Networks

To obtain a $g_{\boldsymbol{w}}(\mathbf{x})$ that closely resembles the ideal mapping g^* between the input and output space, one needs to find suitable parameters $\boldsymbol{w}$. The first step is to pick a differentiable cost or loss function $\mathcal{L}(g_{\boldsymbol{w}}(\mathbf{x}))$ that evaluates the output of the neural network. Subsequently, we try to find the parameters that minimize/maximize $\mathcal{L}(g_{\boldsymbol{w}}(\mathbf{x}))$ (Goodfellow et al., 2016).

Commonly, the mean squared error (MSE) is used in regression settings and the cross-entropy loss in classification settings. Here the labels are included in the calculation of the loss and we could write $\mathcal{L}(g_{\boldsymbol{w}}(\mathbf{x}), \mathbf{y})$. However, in unsupervised learning, loss functions are used that do not depend on labels and we thus stick to the more general notation $\mathcal{L}(g_{\boldsymbol{w}}(\mathbf{x}))$. Moreover, we simplify the notation and write $\mathcal{L}(\boldsymbol{w}; \mathbf{x})$. The expected value of the loss over the data $\mathcal{D}$ is used whenever applicable:

$$\mathcal{L}(\boldsymbol{w}; \mathcal{D}) = \mathbb{E}_{\mathbf{x} \sim p(\mathcal{D})} \left[\mathcal{L}(\boldsymbol{w}; \mathbf{x}) \right]. \tag{2}$$

The parameters that minimize eq. (2) can be (approximately) found with a (stochastic) gradient descent algorithm. The latter iteratively adjusts the parameters $\boldsymbol{w}$ in the opposite direction of the gradient of the loss $\nabla \mathcal{L}(\boldsymbol{w}; \mathbf{x})$. The update step for one data point can be formally written as:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \, \nabla \mathcal{L}(\boldsymbol{w}^{(t)}; \mathbf{x}) \tag{3}$$

where $\boldsymbol{w}^{(t)}$ is the parameter vector at the current time step, $\boldsymbol{w}^{(t+1)}$ denotes the parameter vector at the next time step and η is the learning rate, that determines the step size during each iteration. Training a neural network then consists of two steps that are repeated until some criteria is met: (1) pass some data through the model to calculate the loss, (2) use a form of gradient descent to update the weights. Hereby, the first step is often called a forward pass and the second a backward pass. One iteration over the entire dataset is called an epoch. While this introduces the basic mechanism behind gradient descent in practice many challenges exit, such as increasing the speed of the algorithm or finding the best parameter(s) η, for which more sophisticated implementations have been developed (Robbins, 1951, Kingma and Ba, 2015, Goodfellow et al., 2016).

2.1.3 Bayesian Neural Networks

Being Bayesian in a neural network means that we do not only learn a point estimate for the weights, but infer a posterior distribution (Jospin et al., 2022, MacKay, 1992, Neal, 1992). By Bayes Theorem this distribution is defined as:

$$p(\boldsymbol{w} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{w}) \, p(\boldsymbol{w})}{p(\mathcal{D})} = \frac{p(\mathcal{D} \mid \boldsymbol{w}) \, p(\boldsymbol{w})}{\int_{\boldsymbol{w}'} p\left(\mathcal{D} \mid \boldsymbol{w}'\right) p\left(\boldsymbol{w}'\right) d\boldsymbol{w}'} \tag{4}$$

where $p(\mathcal{D} \mid \boldsymbol{w})$ is the likelihood of the data under our model and $p(\boldsymbol{w})$ is a prior distribution over the weights that can be designed to introduce problem-specific prior knowledge into the network. Training the network results in posterior distributions over the weights that reflect the uncertainty given the observed data. As a consequence, when we sample the

weights of a Bayesian Neural Network (BNN) and input some data we obtain a sample from the related output distribution. By repeated sampling, we can thus estimate relevant uncertainty measures for our predictions. If the output of our neural network for a new data point is $\mathbf{y}$, then the so-called predictive distribution is:

$$p(\mathbf{y} \mid \mathcal{D}) = \int_{\boldsymbol{w}} p(\mathbf{y} \mid \boldsymbol{w})\, p(\boldsymbol{w} \mid \mathcal{D})\, d\boldsymbol{w} = \mathbb{E}_{p(\boldsymbol{w}|\mathcal{D})}[p(\mathbf{y} \mid \boldsymbol{w})] \tag{5}$$

where $p(\mathbf{y} \mid \boldsymbol{w})$ is the probability distribution that arises from passing the data point through the neural network. This probabilistic approach to machine learning comes with a cost as the posterior distribution is usually intractable and has to be approximated. Traditionally, this is done by either Markov Chain Monte Carlo (MCMC) methods or variational inference (VI) methods (Neal et al., 2011, Graves, 2011, Blundell et al., 2015). The first, are computationally expensive but provide asymptotically exact inference, while the second are faster but are not guaranteed to converge to the true posterior. More recently, other cheap approximations of the posterior such as dropout or Laplace inference have also been shown to yield good performance (Gal and Ghahramani, 2016, Daxberger et al., 2021).

2.1.4 Variational Inference in Bayesian Neural Networks

In variational inference, we model the posterior distribution of the weights of a neural network $p(\boldsymbol{w} \mid \mathcal{D})$ with a (tractable) parameterized distribution $q_{\boldsymbol{\theta}}(\boldsymbol{w})$. To find parameters $\boldsymbol{\theta}$ under which the variational distribution resembles the posterior, we minimize the KL divergence:

$$\begin{aligned} KL\left[q_\theta(\boldsymbol{w}) \| p(\boldsymbol{w} \mid \mathcal{D})\right] &:= \int_{\boldsymbol{w}} q_\theta(\boldsymbol{w}) \log \frac{q_\theta(\boldsymbol{w})}{p(\boldsymbol{w} \mid \mathcal{D})}\, d\boldsymbol{w} \\ &= \mathbb{E}_{q_\theta(\boldsymbol{w})}\left[\log q_\theta(\boldsymbol{w}) - \log p(\boldsymbol{w} \mid \mathcal{D})\right]. \end{aligned} \tag{6}$$

Nevertheless, doing this directly is impossible as we are not able to access the true posterior. Instead we maximize a surrogate objective called the evidence lower bound (ELBO). To derive it, we first rewrite the KL divergence as

$$\begin{aligned} KL\left(q_{\boldsymbol{\theta}}(\boldsymbol{w}) \| p(\boldsymbol{w} \mid \mathcal{D})\right) &= \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}\left[\log q_{\boldsymbol{\theta}}(\boldsymbol{w})\right] - \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}[\log p(\boldsymbol{w} \mid \mathcal{D})] \\ &= \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}\left[\log q_{\boldsymbol{\theta}}(\boldsymbol{w})\right] - \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}[\log p(\boldsymbol{w}, \mathcal{D}) - \log p(\mathcal{D})] \\ &= \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}\left[\log q_{\boldsymbol{\theta}}(\boldsymbol{w}) - \log p(\boldsymbol{w}, \mathcal{D})\right] + \log p(\mathcal{D}) \\ &= \log p(\mathcal{D}) - \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}\left[\log \frac{p(\boldsymbol{w}, \mathcal{D})}{q_{\boldsymbol{\theta}}(\boldsymbol{w})}\right]. \end{aligned} \tag{7}$$

If we rewrite the KL divergence like this it is easy to note that only the second part depends on parameters that we can optimize. Thus by dropping the first term and averaging over

the data we get the surrogate objective that we want to maximize:

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) := \mathbb{E}_{\mathcal{D}, q_{\boldsymbol{\theta}}(\boldsymbol{w})} \left[\log \frac{p(\boldsymbol{w}, \mathcal{D})}{q_{\boldsymbol{\theta}}(\boldsymbol{w})} \right]. \tag{8}$$

To see that we have access to $p(\boldsymbol{w}, \mathcal{D})$ and to gain some additional insight into what is happening when a Bayesian Neural Network is trained with VI, rewrite the loss function as:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) &= \mathbb{E}_{\mathcal{D}, q_{\boldsymbol{\theta}}(\boldsymbol{w})} \left[\log p(\boldsymbol{w}, \mathcal{D}) - \log q_{\boldsymbol{\theta}}(\boldsymbol{w}) \right] \\ &= \mathbb{E}_{\mathcal{D}, q_{\boldsymbol{\theta}}(\boldsymbol{w})} \left[\log p(\mathcal{D} \mid \boldsymbol{w}) + \log p(\boldsymbol{w}) - \log q_{\boldsymbol{\theta}}(\boldsymbol{w}) \right] \\ &= \mathbb{E}_{\mathcal{D}, q_{\boldsymbol{\theta}}(\boldsymbol{w})} \left[\log p(\mathcal{D} \mid \boldsymbol{w}) \right] + \mathbb{E}_{q_{\boldsymbol{w}}(\boldsymbol{w})} \left[\log \frac{p(\boldsymbol{w})}{q_{\boldsymbol{\theta}}(\boldsymbol{w})} \right] \\ &= \mathbb{E}_{\mathcal{D}, q_{\boldsymbol{\theta}}(\boldsymbol{w})} \left[\log p(\mathcal{D} \mid \boldsymbol{w}) \right] - KL\left(p(\boldsymbol{w}) \| q_{\boldsymbol{\theta}}(\boldsymbol{w}) \right). \end{aligned} \tag{9}$$

From this, it can be seen that we maximize an empirical term (the Log-likelihood), that depends on the data, while keeping the KL divergence between the prior and the variational distribution as small as possible. To optimize the (negative) objective with gradient descent we need to compute the gradients of the parameters. Unfortunately, this is not a simple task as we cannot move the gradient operator w.r.t. the parameters inside the expectation over the weights. To get around this the so-called reparametrization trick can be used which moves the uncertainty from the parameters into an independent noise variable (Kingma et al., 2015, Blundell et al., 2015).

2.2 Processing Graph-Structured Data with Neural Networks

In the following section the basic concepts behind Graphs are defined and an introduction to processing graph-structured data with Neural Networks is provided. We approach the material from a so-called spectral point of view and roughly follow Isufi et al. (2022). The spectral perspective provides a mathematically thorough underpinning and a connection to the frequency domain. The latter will be used in the second part of the thesis and introducing it now will keep the document consistent. The other perspective on GNNs is called spatial and a comparison between the two can be found in (Zhang et al., 2019). To read the following section, an understanding of Neural Networks and their optimization is required.

2.2.1 Graphs

A graph $\mathcal{G}$ is a mathematical object that consists of a set of N nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$ that connect pairs of nodes. The weights of the edges of a graph are stored in an additional

weight matrix $\mathbf{W}$. For an undirected graph, the neighbourhood of a node i consists of all the nodes that are are connected to it via an edge. Formally, we can write this as

$$\mathcal{N}_i = \{j \in \mathcal{V} : (i,j) \in \mathcal{E}\}. \tag{10}$$

The structure of $\mathcal{G}$ is usually represented in form of a generic matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$, called the Graph Shift Operator (GSO), whose only required property is that

$$[\mathbf{S}]_{ji} = s_{ji} = 0 \text{ whenever } (i,j) \notin \mathcal{E} \text{ for } i \neq j. \tag{11}$$

Frequently used Graph Shift Operators are the adjacency matrix $\mathbf{A}$ with nonzero elements $[\mathbf{A}]_{ji} = a_{ji} = \mathbf{W}_{(i,j)} > 0$ and the Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$. In the latter case, $\mathbf{D} = diag(\mathbf{A1})$ is the degree matrix that contains the degree of each node on the diagonal. We represent data observed on top of a graph, such as node features, in form of a graph signal vector $\mathbf{x} \in \mathbb{R}^N$ where the i th entry $[\mathbf{x}]_i = x_i$ is the value at node i. Depending on the situation, different graph-based tasks are performed using this data. The most-frequent ones that include lables are:

- **Node Classification:** In this task, labels of a subset of nodes are given and we attempt to predict the labels of the unlabeled ones. An example could be the inference of the topic of papers in a citation network.
- **Graph Classification:** This task focuses on predicting a label for an entire graph structure. Examples are the categorization of proteins or the classification of images (in graph form).
- **Signal Classification:** Here, several different signals are given on a common underlying graph structure. The task is then to classify these signals into different categories. An example is the categorization of different brain activity states in a patient. The underlying brain network remains the same for all states, while the activity changes.

2.2.2 Graph Convolutional Filters

To process graph-structured data, Graph Convolutional Filters (GCFs) can be used. A GCF is based on the shift operator $\mathbf{S}$ and operates on the node feature values $\mathbf{x} \in \mathbb{R}^N$. In its most general form it can be written as:

$$\mathbf{z} = \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x} \tag{12}$$

with $\mathbf{h} = [h_0, \dots, h_K]^T$ being the filter parameters and $\mathbf{z} \in \mathbb{R}^N$ being the transformed node values. The locality of such a filter depends on the order K and multiplying the GSO once with the graph signal spreads the information in the features by at most 1-hop. To illustrate this, if the shift operator was the binary adjacency matrix, without self-loops, and we would multiply it it with $\mathbf{x}$, then the output for a specific node would be the sum of the features of the neighbouring nodes (Figure 3).

For a spectral interpretation of this filter the eigendecomposition of $\mathbf{S} = \mathbf{U}\,\mathrm{diag}(\boldsymbol{\lambda})\mathbf{U}^{-1}$ can be leveraged to define a Graph Fourier Transform (GFT) of the signal $\mathbf{x}$, which is defined as

$$\tilde{\mathbf{x}} = \mathbf{U}^{-1}\mathbf{x}. \tag{13}$$

This GFT of a graph signal can be interpreted as a projection onto the eigenspace of the Graph Shift Operator $\mathbf{S}$. Then we can use eq. (13) to map the filter into the frequency domain :

$$\tilde{\mathbf{z}} = \mathbf{U}^{-1}\mathbf{z} = \sum_{k=0}^{K} h_k\,\mathrm{diag}\left(\boldsymbol{\lambda}^{\odot k}\right)\tilde{\mathbf{x}} \tag{14}$$

with $\boldsymbol{\lambda}^{\odot k} \in \mathbb{C}^N : \left[\boldsymbol{\lambda}^{\odot k}\right]_i := \lambda_i^k$ (details in appendix A). The value at index i then turns out to be

$$\tilde{z}_i = \sum_{k=0}^{K} h_k \lambda_i^k \tilde{x}_i = \hat{\mathrm{h}}(\lambda_i)\tilde{x}_i \tag{15}$$

with $\tilde{h}(\lambda) = \sum_{k=1}^{K} h_k \lambda^k$. Thus, the graph convolutional filter in the graph domain, can be seen as a pointwise multiplication in the frequency domain (reminiscent of the convolutional theorem). Furthermore, what we multiply the (fourier-transformed) signal with, is the frequency-response of the filter $\tilde{h}(\lambda)$. In that sense, learning the filter parameters can be seen as optimizing for a specific frequency-response.

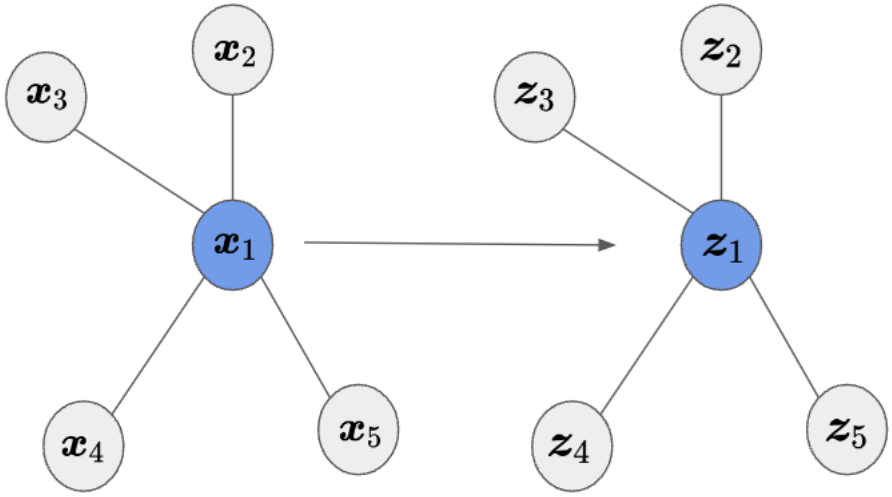

Figure 3: In this example 5 values on the nodes are observed, i.e $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5]$. For a filter $\mathbf{z} = \mathbf{A}\mathbf{x}$ ($K = 1$, $h_0 = 1$) without self-loops we would have $z_1 = x_2 + x_3 + x_4 + x_5$. This filter is local in the sense that each node only receives information from its immediate neighbours. Analysing and designing filters in the graph domain as being done in this figure is called the spatial perspective, as alluded to in the introduction to this section.

2.2.3 Graph Convolutional Neural Networks

Graph Convolutional Neural Networks (GCNNs) consist of multiple Graph Convolutional Filters that are combined to be able to learn a larger variety of functions. Most of the popular GCNN architectures stack GCFs layer-wise and place a nonlinear activation function σ in between each layer. The output at each layer l can then be written as:

$$\mathbf{Z}_\ell = \sigma\left(\sum_{k=0}^{K} \mathbf{S}^k \, \mathbf{Z}_{\ell-1}\mathbf{H}_{\ell k}\right) \tag{16}$$

here $\mathbf{Z}_\ell \in \mathbb{R}^{N \times F_l}$ is a matrix that can contain multiple features for each node, the exact number F_l depends on the layer. Correspondingly, we also have a matrix $\mathbf{H}_{\ell k}$ for the lth layer that contains the kth filter parameters. To give a specific example of such an architecture, let us consider the Graph Convolutional Network (GCN) introduced by Kipf and Welling (2017). In their paper the authors choose $\mathbf{S} = \mathbf{D}^{-1/2}(\mathbf{I} + \mathbf{A})\mathbf{D}^{-1/2}$, $K = 1$ and $\mathbf{H}_{\ell 0} = 0$. Thus, a 2-layered GCN is defined as:

$$\begin{aligned} \mathbf{Z}_1 &= \sigma\left(\mathbf{D}^{-\frac{1}{2}}(\mathbf{I} + \mathbf{A})\mathbf{D}^{-\frac{1}{2}}\mathbf{X}\mathbf{H}_{11}\right) \\ \mathbf{Z}_2 &= \sigma\left(\mathbf{D}^{-\frac{1}{2}}(\mathbf{I} + \mathbf{A})\mathbf{D}^{-\frac{1}{2}}\mathbf{Z}_1\mathbf{H}_{21}\right) \end{aligned} \tag{17}$$

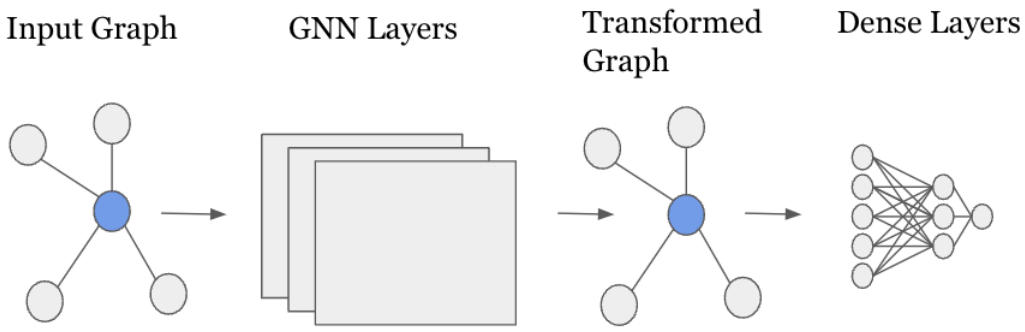

Figure 4: The basic structure of a Graph Neural Network (GNN). Often a number of dense layers are added to transform the output or increase the expressivity of the network. They take the transformed node features as inputs.

where $\mathbf{Z}_2$ contains the processed node features after the second layer. Furthermore, the degree matrices $\mathbf{D}$ are introduced to normalise the adjacency matrix $\mathbf{A}$. The GCN is local in the sense that it only aggregates information from 2-hop neighbours for each node. In a Graph Neural Network, the weights of the network are the filter parameters. However, often additional dense neural network layers are added or a parameterised loss function is used after the GCN layers (Figure 4). Therefore, we will refer to a general weight of a neural network as w. Nevertheless, if we want to specifically emphasize that we are talking about a filter parameter we refer to it as h as in the previous section.

To illustrate how a GCN can be trained in a supervised setting consider a node classification task where we observe labels for V nodes. Then we can use the categorical cross-entropy loss for C classes to optimize the parameters. The output of the GCN would be a set of node embeddings, which are subsequently passed through a softmax layer to generate probabilities for each class. The related categorical cross-entropy loss is:

$$\mathcal{L}_{ce} = -\sum_{n=1}^{V}\sum_{l=1}^{C} \mathbf{y}_{nl} \ln \mathbf{p}_{nl} \tag{18}$$

where $\mathbf{y}_{nl}$ is the ground-truth label, and $\mathbf{p}_{nl}$ is the predicted probability for node n belonging to class l. This objective can the be optimized to train the model.

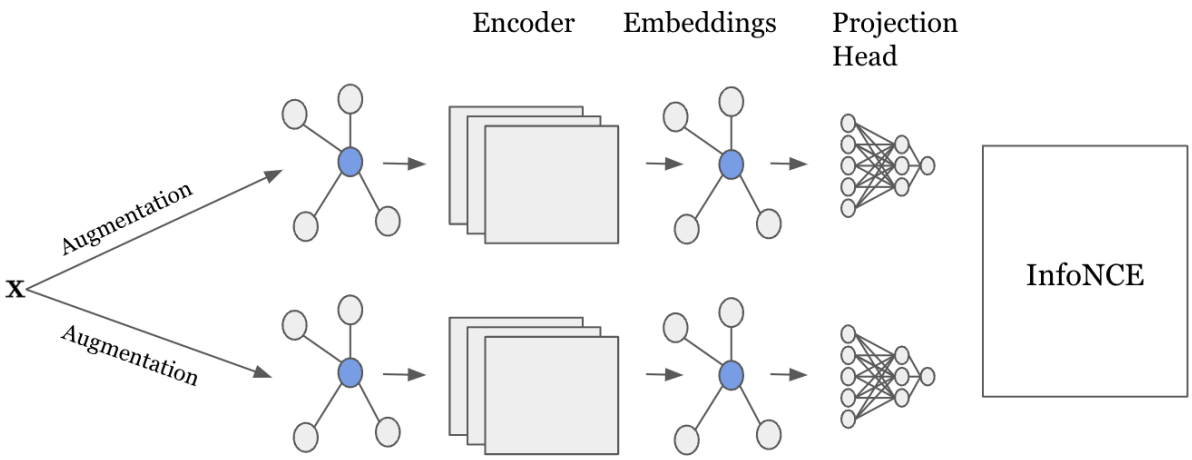

Figure 5: The basic structure of a Graph Contrastive Learner (GCL). First augmentations of the original data point are created which are then both passed through an encoder. Note that usually the same neural network is used as an encoder and often one of the augmentations is the identity augmentation (i.e. the original data point is encoded). Furthermore, this figure includes a projection head, which is usually discarded after training.

2.3 Contrastive Learning with the InfoNCE

The InfoNCE loss is an unsupervised objective and aims to train a neural network such that it maps similar (positive) data points close to each other in an embedding space (van den Oord et al., 2018). Embeddings generated in this way capture important latent information of the data and provide a compact and meaningful representation. Importantly, the embeddings exist in the euclidean space and can be used as features in a variety of downstream tasks (e.g. classification or clustering). For images an graphs, the positive examples of an existing data point (anchor) are usually generated by using similarity-preserving data augmentation methods.

2.3.1 The InfoNCE as a contrastive loss

In contrastive learning we encode a data point $\mathbf{x}$ and its positive example $\mathbf{x}'$ with encoders Q_{ϕ_1}, Q_{ϕ_2} (typically we share weights and use the same neural network, i.e. $\phi = \phi_1 = \phi_2$) to obtain the embeddings $\mathbf{z} = Q_{\phi_1}(\mathbf{x})$ and $\mathbf{z}' = Q_{\phi_2}(\mathbf{x}')$. To optimize the network parameters we then minimize the InfoNCE loss:

$$\mathcal{L} = -\mathop{\mathbb{E}}_{X}\left[\log \frac{f_{\rho}\left(\mathbf{z}, \mathbf{z}'\right)}{f_{\rho}\left(\mathbf{z}, \mathbf{z}'\right) + \sum_{j=1}^{M} f_{\rho}\left(\mathbf{z}, \mathbf{z}_j'\right)}\right] \tag{19}$$

where $f_{\rho}(\mathbf{z}, \mathbf{z}')$ is a similiarity function (usually the cosine similarity) with parameters ρ and M (negative) examples are randomly sampled from the data. Optimizing this loss encourages the model to map the positive data points in the numerator close to each other in the embedding space while the negative ones are pushed further apart. In fact, learning the InfoNCE maximizes a lower bound on the mutual information $I(\mathbf{x}; \mathbf{z}')$ (derivation in Appendix C). This means, that we train a model to generate embeddings that (approximately) contain the shared information between $\mathbf{x}'$ and $\mathbf{x}$. As a consequence, if we design data augmentations that destroy the irrelevant information the relevant parts will be encoded. A more detailed discussion about the origin of the InfoNCE is deferred to Appendix B.

2.3.2 Contrastive Learning in Practice

The general framework of the InfoNCE introduced in the previous section is highly adaptable and has been shown to be effective in generating useful embeddings for images and graphs (Jaiswal et al., 2021, Balestriero et al., 2023, Xie et al., 2023). In these applications one challenge is that in practice unlabled data is used and true positives are usually not available. An efficient work-around are similarity-preserving data augmentation techniques that generate (approximate) positive samples. For instance, on images this could be changing the colour scheme of a data point (colour jittering) and it is useful when the latent is invariant to the colour. For example, if we wanted to differentiate between cats and dogs, then a brown cat should be mapped to a similar place in the embedding space as a white cat. We want to preserve the information that is not the colour and can apply colour jittering. For graphs, common augmentations are edge dropping, feature masking or subgraph sampling. To illustrate why these are useful consider a graph with scientific papers as nodes, words as features and citation links as edges. If the task is to predict a subject label for each paper (node) then it might be more relevant which other papers they cite and not which words they contain. Consequently, a feature masking augmentation could be a good choice so that nodes with many common citations are mapped to the same place, even if the used words are different.

Many works that use the InfoNCE on graphs and images use a dual-branch architecture that creates two augmentations and compare them to each other, see Figure 5 (e.g., Chen et al., 2020, Zhu et al., 2020, You et al., 2020). Varying the augmentation method used in each of the branches can improve the performance of the learner. Furthermore, many popular works alter the formulation of the loss function and the used architecture. Importantly, in this thesis we follow SimCLR (Chen et al., 2020) who popularized an additional

temperature term τ and use the similarity function:

$$f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right) = \exp\left(\frac{csim(\mathbf{z}, \mathbf{z}')}{\tau}\right) \tag{20}$$

where $csim(\mathbf{z}^T\mathbf{z}') = \frac{\mathbf{z}^T\mathbf{z}'}{\|\mathbf{z}\|\|\mathbf{z}'\|}$. They also introduced a parametric map (the projection head) $g_{\boldsymbol{\rho}}(\cdot)$ from the embeddings to the space where the loss is calculated (Figure 5). For theoretical derivations we can absorb it into the similarity function:

$$f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right) = \exp\left(\frac{csim(g_{\boldsymbol{\rho}}(\mathbf{z}), g_{\boldsymbol{\rho}}(\mathbf{z}'))}{\tau}\right). \tag{21}$$

We denote the representations inside the space in which the loss is calculated as $\mathbf{h} = g_{\boldsymbol{\rho}}(\mathbf{z})$ and $\mathbf{h}' = g_{\boldsymbol{\rho}}(\mathbf{z}')$.

3 Uncertainty in Contrastive Learning

Despite the success of the InfoNCE for learning embeddings, research into probabilistic approaches and uncertainty estimation is rare. A notable exception is the work of Aitchison and Ganev (2023), which proposes a generative model for contrastive learning and derives an evidence lower bound that can be used to train the encoder. In this chapter, we extend this result to include distributions over the parameters of the model which naturally gives rise to a notion of (Bayesian) epistemic uncertainty for contrastive learning. We implement this approach by training a Bayesian Graph Neural Network with Gaussian priors as an encoder and make the empirical finding that regularizing the approximating variational family further improves the accuracy on downstream classification tasks.

Building up upon this, we propose a new uncertainty measure for contrastive learning based on the disagreement in likelihood due to different positive examples. In contrast to existing uncertainty measures, it has a principled probabilistic interpretation and incorporates the epistemic uncertainty in the weights when using a BNN. The proposed measure is inspired by research on Bayesian Variational Autoencoders that uses a reconstruction loss (Daxberger and Hernández-Lobato, 2019). We thus show how the probabilistic interpretation of the InfoNCE can be used in practice to solve problems in contrastive learning with methods inspired by the literature on generative models. This chapter is based on the recent publication *Uncertainty in Graph Contrastive Learning with Bayesian Neural Networks* (Möllers et al., 2023).

3.1 Variational Graph Contrastive Learning (VGCL)

3.1.1 Weight Uncertainty for Contrastive Learning

In the probabilistic contrastive learning model (Aitchison and Ganev, 2023) we observe two correlated data points $\mathbf{x}$ and $\mathbf{x}'$ generated by two latent variables $\mathbf{z}$, $\mathbf{z}'$. In this setting, the likelihood is the joint distribution $p\left(\mathbf{x}, \mathbf{x}'\right)$. The mapping between the data and the latents is approximated with two probabilistic encoders $\mathrm{Q}(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\phi}_1)$, $\mathrm{Q}(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\phi}_2)$, usually $\boldsymbol{\phi} = \boldsymbol{\phi}_1 = \boldsymbol{\phi}_2$. To be Bayesian, we then learn distributions over the weights $\boldsymbol{w} = (\boldsymbol{\phi}, \boldsymbol{\rho})$, where $\boldsymbol{\rho}$ parameterizes the similarity function. We approximate these with a a Gaussian variational family and pick Gaussian prior distributions, i.e. $p(\boldsymbol{w}) = \prod_j \mathcal{N}\left(w_j \mid \mu, \sigma^2\right)$ and $q(\boldsymbol{w} \mid \boldsymbol{\theta}) = \prod_j \mathcal{N}\left(w_j \mid \theta\right)$ where each w_j is a weight of either the encoder or the projection head. Furthermore, following Blundell et al. (2015), we reparameterize the standard deviation of the variational distribution as $\sigma_v = \log(1 + \exp(a))$ and the parameter set is thus $\theta = (\mu_v, a)$. The parameters for this probabilistic model are then optimized by maximizing

the ELBO (Appendices D and E):

$$\log p\left(\mathbf{x}, \mathbf{x}'\right) \geq \mathbb{E}_{\mathrm{Q}(\mathbf{z},\mathbf{z}'|\mathbf{x},\mathbf{x}',\boldsymbol{w})\mathrm{q}(\boldsymbol{w}|\boldsymbol{\theta})}\left[\log \frac{f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right)}{f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right)+\sum_{j=1}^{M} f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}_j'\right)}\right]-KL(\mathrm{q}(\boldsymbol{w} \mid \boldsymbol{\theta}) \,\|\, p(\boldsymbol{w})). \tag{22}$$

The first part of this equation is, for a deterministic encoder and averaging over the data, equivalent to the InfoNCE in eq. (19) while the second part encourages the variational family to resemble the prior distribution. During training we sample from $q(\boldsymbol{w} \mid \boldsymbol{\theta})$ and approximate the expectation in eq. (22) with a Monte Carlo estimate.

Importantly, due to how we obtain the data $\mathbf{x}$, $\mathbf{x}'$ (e.g. via an augmentation), the probability of observing two similar (positive) examples together is high. The empirical InfoNCE term encourages the model to map these correlated examples to the same place. From the generative model perspective, this leads to a mapping such that examples generated from similar latents have a high probability of appearing together.

3.1.2 Regularizing the Variational Family in Bayesian Contrastive Learning

In the last section we have seen how we can train a Bayesian self-supervised learner with Gaussian Priors. In addition to that, we propose to regularise the variational family to encourage larger variance in the weights. This might be beneficial because we are training the model on augmentations, which are only an approximation to the data that would be the result of a generating process given the true latents. To overfitting on these augmentations we want the model to be more uncertain and incorporate this knowledge by regularizing the variance. We do this by placing Gaussian hyperpriors over the parameters of the variational family:

$$\mu_v \sim \mathcal{N}(0, \sigma_0^2)\,,\; a \sim \mathcal{N}(\mu_a, \sigma_a^2) \tag{23}$$

Hereby, we can place a Gaussian over a as it can be both negative or positive. We approximate these hyperpriors with their MAP estimates. We refer to the resulting approach as Variational Graph Contrative Learning (VGCL) in what is to follow.

3.2 The Contrastive Model Disagreement Score (CMDS)

3.2.1 Introducing the CMDS

Now that we can train a contrastive learner with a probabilistic encoder we want to equip our model with uncertainty estimates that help us predict downstream performance. Existing work focuses on the embedding uncertainty of encoding one data point $\mathbf{x}$, but the

likelihood $p(\mathbf{x}, \mathbf{x}')$ is based on two data points and is approximated with the contrastive loss. Our uncertainty measure should thus be related to this likelihood and additionally include the epistemic uncertainty of the weight distribution when using a BNN. To incorporate the contrastive nature of the problem, we propose to quantify the uncertainty of $\mathbf{x}$ by using the variation of $p(\mathbf{x} \mid \mathbf{x}')$ under samples $\mathbf{x}' \sim p(\mathbf{x}' \mid \mathbf{z}')$ that are generated by augmenting the original data point. If the model has learned a coherent explanation for observing $\mathbf{x}$ in this context, then the likelihood it assigns for different positive examples should be similar.

A related idea exists in the literature on Bayesian Variational Autoencoders (BVAE). In that setting, Daxberger and Hernández-Lobato (2019) measure the disagreement in the likelihood estimates of different models sampled from a learned weight distribution. The intuition hereby is that for a data point with low uncertainty the models should agree in their prediction. Analogously, in contrastive learning we can sample likelihood estimates by drawing different positive samples. Combining this with samples from the weight distributions we obtain the Contrastive Model Disagreement Score (CMDS):

$$D_{\text{CMDS}}(\mathbf{x}) = \frac{1}{\sum_{t=1}^{M} l_t(\mathbf{x})^2} \qquad \text{with} \quad l_t(\mathbf{x}) = \frac{p\left(\mathbf{x} \mid \mathbf{x}_t', \boldsymbol{\phi}_t, \boldsymbol{\rho}_t\right)}{\sum_{i=1}^{M} p\left(\mathbf{x} \mid \mathbf{x}_i', \boldsymbol{\phi}_i, \boldsymbol{\rho}_i\right)}, \tag{24}$$

where we generate M samples from $\mathbf{x}_t' \sim p(\mathbf{x} \mid \mathbf{z}')$ via augmentations and $\boldsymbol{\rho}_t, \boldsymbol{\phi}_t \sim q(\boldsymbol{w} \mid \boldsymbol{\theta})$ from the learned variational families. This score measures the disagreement between the different sampled likelihood estimates. It is small when the normalised likelihood are different from each other and large when they are alike. In contrast to the measure proposed by Daxberger and Hernández-Lobato (2019), the CMDS can be used for deterministic encoders and it is thus not limited to Bayesian Neural Network approaches.

3.2.2 Mathematical Interpretation of the CMDS

The first step when calculating the CMDS for a data point $\mathbf{x}$ is to normalise the M different sampled likelihoods. By normalising we ensure that the variation between the likelihoods is independent of the absolute size. Otherwise, large variations would often occur for data points with high likelihoods which would make the measure unsuitable for our purposes. From the normalisation step we obtain the set $\{l_t(\mathbf{x})\}_{t=1}^{M}$ which we can the use to calculate the score $D_{\text{CMDS}}(\mathbf{x}) \in [1, M]$. This measure of uncertainty, assumes its maximum when the sampled likelihoods are uniformly distributed. On the contrary it will be small when the likelihoods are different from each other, with a few values being comparably large. In the extreme case, it is equal to 1. This happens when we have $\{l_t(\mathbf{x})\}_{t=1}^{M} = \{0, ..., 0, 1, 0, ..., 0\}$.

To position the CMDS in the field of Bayesian Statistics, let us assume that we have observed some data $\mathcal{D}$, then the posterior distribution over the weights of the encoder and projection head is:

$$p(\boldsymbol{w} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})} = \frac{p(\mathcal{D} \mid \boldsymbol{w})p(\boldsymbol{w})}{\int_{\boldsymbol{w}'} p\left(\mathcal{D} \mid \boldsymbol{w}'\right) p\left(\boldsymbol{w}'\right) d\boldsymbol{w}'}. \tag{25}$$

Suppose we infer this distribution and subsequently observe a new pair of data points $\mathbf{x}, \mathbf{x}'_{new}$. Then the new posterior for the weights based on these new data points is:

$$p(\boldsymbol{w} \mid \mathcal{D}^*) = \frac{p(\mathbf{x}, \mathbf{x}'_{new} \mid \boldsymbol{w})\, p(\boldsymbol{w} \mid \mathcal{D})}{p(\mathbf{x}, \mathbf{x}'_{new} \mid \mathcal{D})} \tag{26}$$

where $\mathcal{D}^* = \mathcal{D} \cup \{\mathbf{x}, \mathbf{x}'_{new}\}$. We can thus write:

$$p(\boldsymbol{w} \mid \mathcal{D}^*) = lp(\boldsymbol{w} \mid \mathcal{D}) \tag{27}$$

with $l = \frac{p(\mathbf{x}, \mathbf{x}'_{new} \mid \boldsymbol{w})}{p(\mathbf{x}, \mathbf{x}'_{new} \mid \mathcal{D})}$ being the normalised likelihood. l can be interpreted as the change between the the original posterior and the new posterior after observing the additional data points. Taking the finite sample estimator shows that this directly relates to the quantity l_t used in the CMDS:

$$\begin{aligned} l &= \frac{p(\mathbf{x}, \mathbf{x}'_{new} \mid \boldsymbol{w})}{p(\mathbf{x}, \mathbf{x}'_{new} \mid \mathcal{D})} \\ &= \frac{p\left(\mathbf{x}, \mathbf{x}'_{new} \mid \boldsymbol{w}\right)}{\mathbb{E}_{p(\boldsymbol{w} \mid \mathcal{D})}\left[p\left(\mathbf{x}, \mathbf{x}'_{new} \mid \boldsymbol{w}\right)\right]} \\ &\simeq \frac{p\left(\mathbf{x}, \mathbf{x}'_{new} \mid \boldsymbol{w}\right)}{\frac{1}{M}\sum_{i=1}^{M} p\left(\mathbf{x}, \mathbf{x}'_{new} \mid \boldsymbol{w}_i\right)} \\ &= M l_t. \end{aligned} \tag{28}$$

To summarize, we have established a link between the new and the old posterior and then we have shown that the quantity l that connects the two can be proportionally estimated by the quantity l_t used in the CMDS. Next, note that expectation for any function $g_{\boldsymbol{w}}(\mathbf{x}, \mathbf{x}'_{new})$ (could be a neural network) under $p(\boldsymbol{w} \mid \mathcal{D}^*)$ can be estimated by using:

$$\mathbb{E}_{p(\boldsymbol{w} \mid \mathcal{D}^*)}[g_{\boldsymbol{w}}(\mathbf{x}, \mathbf{x}'_{new})] \simeq \mathbb{E}_{p(\boldsymbol{w} \mid \mathcal{D})}\left[M l_t g_{\boldsymbol{w}}(\mathbf{x}, \mathbf{x}'_{new})\right] \simeq \sum_{t=1}^{M} l_t g_{\boldsymbol{w}_t}(\mathbf{x}, \mathbf{x}'_{new}). \tag{29}$$

Here we are doing importance sampling with respect to the proposal distribution $p(\boldsymbol{w} \mid \mathcal{D})$ and with importance weights l_t. This estimate depends on how well the proposal distribution $p(\boldsymbol{w} \mid \mathcal{D})$ fits the distribution $p(\boldsymbol{w} \mid \mathcal{D}^*)$. In the literature for importance sampling one

well-known measure for this is the effective sample size (ESS) and it is the CMDS:

$$ESS = \frac{1}{\sum_{t=1}^{M} l_t^2} = D_{\text{CMDS}} \tag{30}$$

In that sense, the disagreement score can be viewed as a measure that quantifies the change in distribution between $p(\boldsymbol{w} \mid \mathcal{D})$ and $p(\boldsymbol{w} \mid \mathcal{D}^*)$, or ,equivalently, the informativeness of observing a new pair of data points $\mathbf{x}$ and $\mathbf{x}'_{new}$. If the new pair of data points is uninformative for the model and induces only a small changes in the weights then the model has learned a coherent explanation and is certain about its prediction.

There are two ways to understand why including additional constrastive samples into the CMDS is beneficial. First, it is an approximation to $\mathbb{E}_{p(\boldsymbol{w}|\mathcal{D}^*)}[\mathbb{E}_{p(\mathbf{x}'_{new}|\mathbf{z}')}[g_{\boldsymbol{w}}(\mathbf{x}, \mathbf{x}'_{new})]]$ which gives a better estimate of uncertainty than simply taking one pair as a sample. For the second, note that we can theoretically shift the probabilities of the augmentations to the first-layer weights of the neural network. Encoding an augmented data point $\mathbf{x}'$, can be seen as encoding $\mathbf{x}$ but with an altered (dropout) distribution in the first layer. In that sense, we can see the procedure of encoding $\mathbf{x}$ and $\mathbf{x}'$ as equivalent to only encoding $\mathbf{x}$ but with two neural networks, one of which has a dropout first layer. When looking at it from that perspective, sampling augmentations can be interpreted as sampling weights in the dropout layer of the model and is coherent with the motivation of the CMDS as a measure for information gain.

4 Experiments

4.1 Set-up

Datasets and Experimental Procedure We evaluate the proposed methods on the Planetoid citation datasets (Cora, Citeseer, Pubmed) for node classification. The nodes of these datasets are scientific publications while the edges are citation links. Moreover, the features of each node are a sparse bag-of-words in form of a $0/1$ vector. If an entry is 1, then the corresponding word appears in the article. Based on the given data, the subject label is inferred for each scientific article. For the Cora dataset we have a total of 2708 labled scientific articles, for Citeseer 3327 and for Pubmed 19717.

In the experiments, we follow a standard transductive set-up for SSL to evaluate the methods (Zhu, Xu, Liu and Wu, 2021). That means, that we train the self-supervised learner on all available unlabeled data points and afterwards fit a l_2-regularized logistic regression on the obtained embeddings. Hereby, we use 10% of the data to train the linear classifier, 10% for validation and 80% for the evaluation. For the Bayesian models the classifier is trained on the average embedding over 100 samples. We repeat this evaluation procedure 20 times and report the mean and standard error estimates. The likelihood is approximated with the contrastive loss whenever required.

Implementation Details We use the same graph neural network to encode both data points and choose a two-layered GCN as in eq. (17) to do so. Similar to Zhu et al. (2020) we choose the hidden layer size to be 128 and use ReLU activation functions. The projection head, consists of two MLP layers with a hidden layer size of 128 and an ELU activation.

Training and Hyperparameters The models are trained for 150 epochs on the Cora and Citeseer datasets and for 1500 epochs on Pubmed. Furthermore, we follow Zhu et al. (2020) and augment the data by masking features and dropping edges. This is done via Bernoulli dropout and we thus tune the related probabilities $p_{f,1}, p_{e,1}, p_{f,2}, p_{e,2}$ to generate two augmentations of the original graph. Following Zhu, Xu, Liu and Wu (2021), we then compare these via the contrastive loss during training. All parameters are optimized with the Adam algorithm (Kingma and Ba, 2015) and as loss function we choose the temperature-scaled InfoNCE as introduced by Chen et al. (2020). For the probabilistic encoders 20 samples are averaged to obtain a loss estimate.

Table 1: Test accuracies for different unsupervised methods on the citation datasets. Our proposed method outperforms all the baselines on all tasks.

Dataset	**Cora**	**Citeseer**	**Pubmed**
Raw features	64.8	64.6	84.8
node2vec	**74.8**	**52.3**	80.3
DeepWalk	**75.7**	**50.5**	80.5
Deep Walk + Features	73.1	47.6	83.7
GAE	76.9	60.6	82.9
VGAE	78.9	61.2	83.0
DGI	82.6 ± 0.4	68.8 ± 0.7	86.0 ± 0.1
InfoNCE	81.9 ± 0.4	70.8 ± 0.1	85.0 ± 0.1
VI-InfoNCE	82.1 ± 0.3	71.0 ± 0.1	85.0 ± 0.1
VGCL (Ours)	$\mathbf{83.5 \pm 0.2}$	$\mathbf{72.2 \pm 0.1}$	$\mathbf{86.3 \pm 0.1}$

Evaluation and Baselines As baselines for the accuracies obtained with our model we consider several existing methods from the literature (see table 1). Among these, the ones with acronyms are the Graph Autoencoder, the Variational Graph Autoencoder (GAE,VGAE, Kipf and Welling, 2017) and Deep Graph Infomax (DGI, Veličković et al., 2019). We implemented the InfoNCE, VI-InfoNCE and VGCL methods and took the results for the other ones from Zhu et al. (2020) where the same datasets and experimental procedure are used as in our work.

In addition to that, we compute a number of unsupervised uncertainty measures and compare their performance to the CMDS. The baselines that we compute are the Per-Sample-Feature-Variation (PSFV, Ardeshir and Azizan, 2022), the Average-Standard-Deviation of the features (ASTD, Hasanzadeh et al., 2021), the expected likelihood under positive samples $\mathbb{E}_{\mathbf{x}' \sim p(\mathbf{x}|\mathbf{z}')} [p(\mathbf{x} \mid \mathbf{x}')]$ and the related Watanabe-Akaike Information Criterion (WAIC) $\mathbb{E}_{\mathbf{x}' \sim p(\mathbf{x}|\mathbf{z}')} [p(\mathbf{x} \mid \mathbf{x}')] - \mathrm{Var}_{\mathbf{x}' \sim p(\mathbf{x}|\mathbf{z}')} [\log p(\mathbf{x} \mid \mathbf{x}')]$.

4.2 Results

In the experiments we observe that the accuracy obtained with a self-supervised learner with variational inference (VI-InfoNCE) roughly matches the performance of the learner trained with deterministic encoders. Nevertheless, the performance of the first can be greatly improved by regularising the variational family. We see that the regularisation

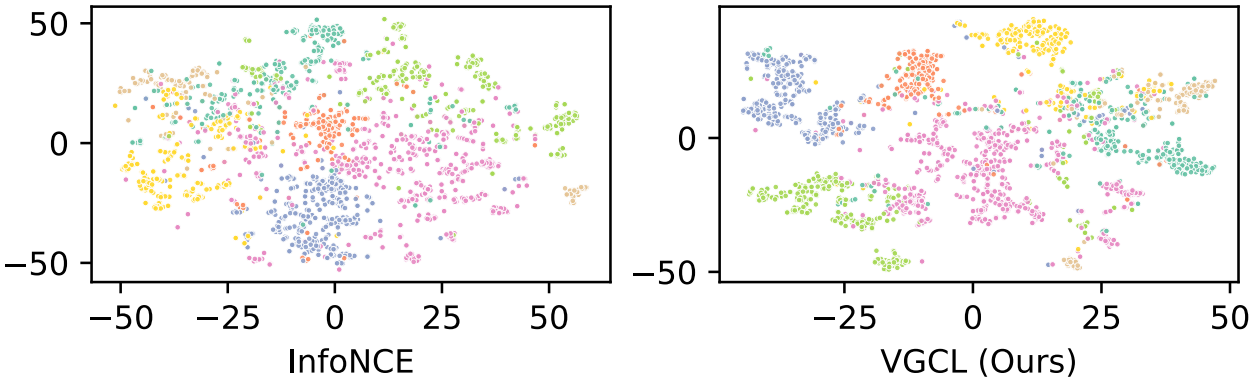

Figure 6: T-SNE plots of the embeddings generated by a deterministic encoder (left) in comparison to the embeddings generated by our probabilistic encoder with hyperpriors (right). Our method generates a better separation between the classes in the embedding spaces, which might explain its improved performance.

indeed leads to larger uncertainty in the weights (Figure 9) and that such a learner outperforms the other methods which were trained with the InfoNCE by up to 1.4 percent points (Table 1). Furthermore, a tsne-plot of the embeddings shows that VGCL seperates the classes better than its deterministic counterpart, which is a likely reasons for the better predictions of the downstream classifier (Figure 6).

For the uncertainty estimates one can see that already for deterministic encoders the CMDS leads to a better ordering of the samples (Figure 7). This ordering can be further improved by introducing uncertainty over the weights. Furthermore, the uncertainty estimates of a Bayesian model with a regularized variational family are better calibrated than the ones obtained with standard variational inference. Interestingly, considering distributions over the weights seems to impact some uncertainty measures more than for others (Figure 8). Hereby, we see larger changes in the measures that are calculated based on the contrastive loss (Likelihood, WAIC, CMDS) and comparably smaller ones in the measures that focus on the embedding space (ASTD, ASTD_Norm, PSVF). One reason for this could be, that the first incorporate the epistemic uncertainty in the projection head, while the second can only make use of the incomplete epistemic uncertainty in the encoders. We have added additional results in appendix I.

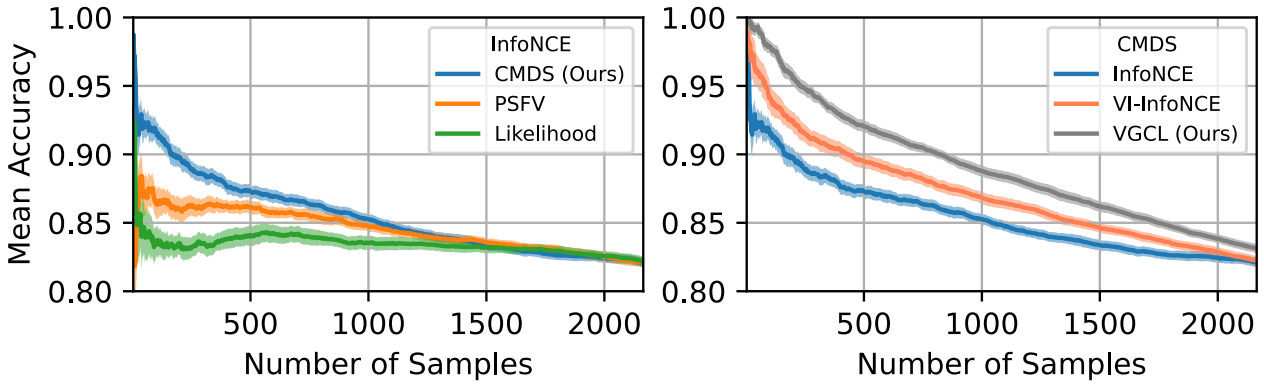

Figure 7: (Left) Error retention curves on Cora for different uncertainty measures for a deterministic model trained with the InfoNCE. The CMDS yields the best sorting. (Right) Performance of the CMDS for probabilistic models (VI-InfoNCE, VGCL). Incorporating Bayesian epistemic uncertainty into the CMDS improves the ordering. The error retention curves are generated by ordering the test embeddings by increasing uncertainty and the mean accuracy is calculated over gradually more samples. For a good uncertainty measure the mean accuracy is high for the most certain samples and decreases as more uncertain data points are included.

5 Discussion

We introduced Variational Graph Contrastive Learning (VGCL) as a probabilistic method for node classification and the Contrastive Model Disagreement Score (CMDS) as a related uncertainty measure. Employing VGCL has led to a significant increase in accuracy on the investigated tasks. Especially, the choice of the prior over the weights and the regularization of the approximating variational family has had a strong impact on the quality of the embeddings. In the experiments, we have shown this improved performance for a standard InfoNCE learning setting from the literature. Despite this first very positive result, we have to be aware that many different ways of generating augmentations, picking encoders or selecting negative samples exist. It is not clear how the obtained results would change when the setting is varied in that way. A further investigation in that direction would be interesting and could also provide insights into whether the hypothesis that regularising the variational family essentially prevents the learner to overfit on imperfect augmentations is valid. For problems for which we can design very accurate augmentations we would then expect the benefits of VGCL to be smaller.

After training a probabilistic contrastive learner the CMDS can be used to obtain an ordering of the data points by uncertainty. Our experiments have shown that this ordering is remarkably effective for predicting the performance in subsequent downstream. This could lead the CMDS to find applications in areas such as active learning (e.g., to mitigate

the cold start problem) or unsupervised out-of-distribution detection. That being said, note that while the CMDS consistently delivered superior unsupervised ordering in all our experiments, it is not a one-size-fits-all solution. The Disagreement Score is not bound to a distribution and serves as a proxy measure for test accuracy in downstream tasks, rather than providing an uncertainty over the actual embedding. The relevance of this distinction becomes clear in a scenario where labels have been collected to train a supervised learner (not only for evaluation) on the embeddings. In such cases, the learner could utilize the uncertainties in the features, potentially improving its own predictions and uncertainty estimates. For instance, by making predictions for different sampled embeddings and measuring the variation in the predictions. The current formulation of the CMDS, which is not being calculated in the embedding space, is not amenable to these types of tasks.

6 Related Work

Most research on uncertainty in contrastive learning focuses on embedding uncertainty. That means, one data point is mapped into the embedding space and then the related uncertainty is quantified there. That can either be done by calculating measures, such as the standard deviation of the embedding features under different augmentations directly, or by using surrogate distributions (e.g. GMM) in the embedding space (e.g. Ardeshir and Azizan, 2022, Wu and Goodman, 2020, Hasanzadeh et al., 2021). Another different approach to estimating uncertainty is by using a proxy score. For instance, Zhang et al. (2021) realize that the temperature parameter of the InfoNCE loss can give an indication about the hardness of an example. They then learn it as an input-dependent variable for each data point which results in an uncertainty score. To the best of our knowledge, none of the existing works quantifies uncertainty with respect to a (unsupervised) likelihood of the data. This stands in contrast to the CMDS, which is constructed upon the probabilistic model proposed by Aitchison and Ganev (2023). Furthermore, we outperform other unsupervised measures of uncertainty such as the ASTD in our experiments.

All that being said, there are a few works that employ Bayesian reasoning for contrastive learning. For instance, Sharma et al. (2023) incorporate unlabeled data into (supervised) Bayesian Neural Networks and use contrastive methods to set better priors. Different to our work, they only do last layer inference and do not consider graphs. Furthermore, Wang and Yang (2022) calculate pseudo-labels and use a Bayesian non-parametric method to learn embeddings that are more robust to noise. While they conduct experiments on graphs, their methods and contributions are mostly unrelated from what is proposed in

this thesis. Moreover, Liu and Wang (2023) propose Bayesian Self-Supervised Contrastive Learning which uses a Bayesian approach to learn parameters in a debiased InfoNCE loss. In this work no inference over the weights of the network is performed. Last but not least Hasanzadeh et al. (2021) use Bayesian inference coupled with Beta-Bernoulli priors and obtain an improved performance by learning the parameters of the augmentations. As mentioned before, they propose the ASTD. In contrast to their work we measure uncertainty with respect to the likelihood of the probabilistic model and our performance improvements stem from a better regularization of the variational family. Moreover, we do full network inference including the projection head.

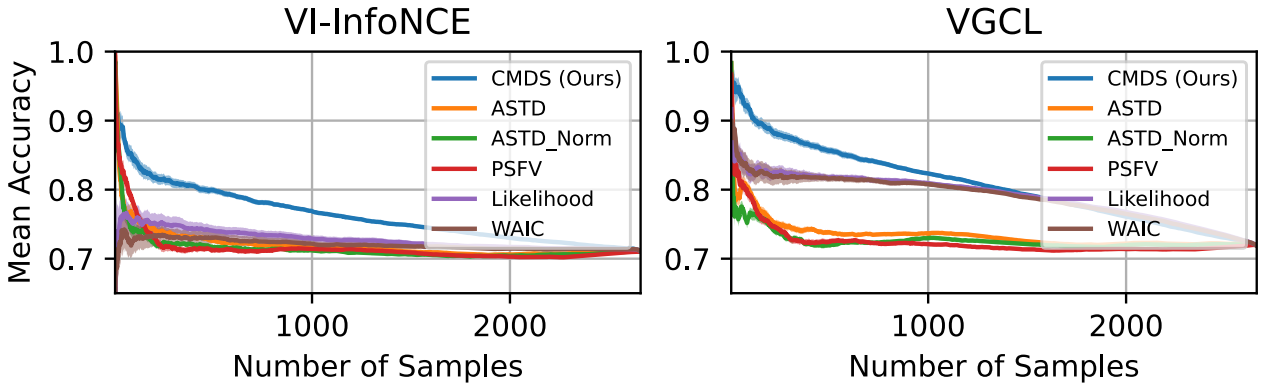

Figure 8: (Left) Retention Curves on CiteSeer for VI-InfoNCE. (Right) Retention Curves on CiteSeer for VGCL.

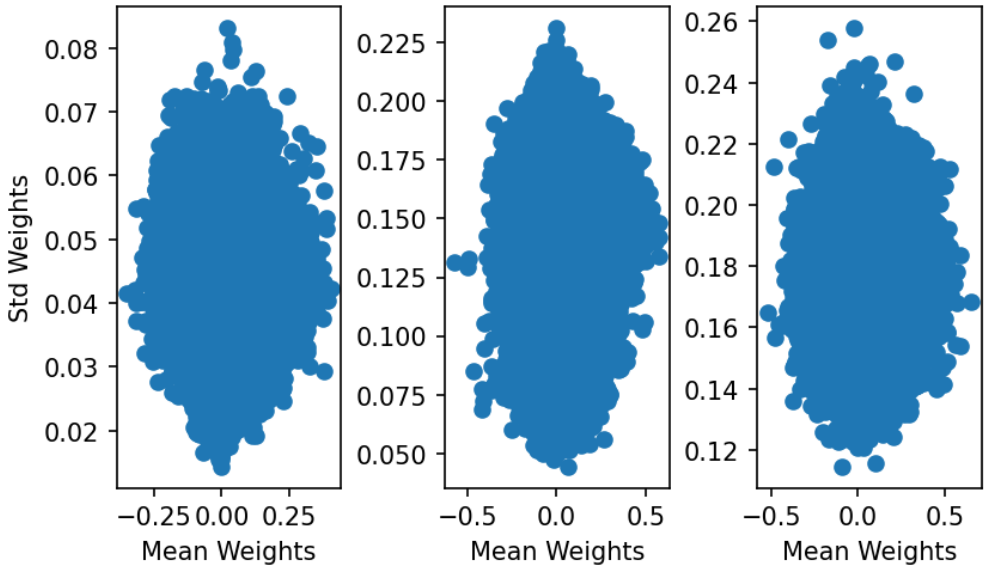

Figure 9: Parameters of the first layer weight distributions of VGCL under different regularizations of the variance of the variational family. To generate this plot we let $a \sim \mathcal{N}(0, \sigma_a^2)$ and $\sigma_a^2 = \{1, 0.001, 0.0000001\}$ from left to right in the figure. At $a = 0$ the variance of the variational family is 0.301, which is very large for networks weights. A stronger prior, i.e. smaller variance, towards this value thus leads to the weight distributions to have larger standard deviations.

Part II

Contrastive Learning on Simplicial Complexes

7 Background II

7.1 Processing Data on Simplicial Complexes

7.1.1 Simplicial Complexes

While graphs provide a powerful framework to represent systems that are made up of interacting entities, their usefulness is limited when modeling data that is not naturally defined on nodes. To make up for this shortcoming simplicial complexes have been proposed as a powerful tool to model edge flows. In the following the basic definitions related to simplicial complexes will be reviewed (Schaub et al., 2021).

Definition 7.1 (p-simplex and simplicial complex)**.** Given a set of vertices $\mathcal{V}$ a p-simplex S^p is a subset of $\mathcal{V}$ of cardinality $p+1$. A simplicial complex S is a set of simplices such that if S^p is in S then any subset of S^p must also be in S.

According to this definition, we can see nodes as 0-simplices, edges as 1-simplices and (filled) triangles as 2-simplices. A co-face of a p-simplex S^p is a $(p+1)$-simplex such that S^p is its subset. Illustrating this, a triangle could be a co-face of an edge that is forming it. A face of a $p+1$-simplex S^{p+1} is a subset of S^{p+1} with cardinality p. So the face of a triangle (2-simplex) could be an edge (1-simplex).

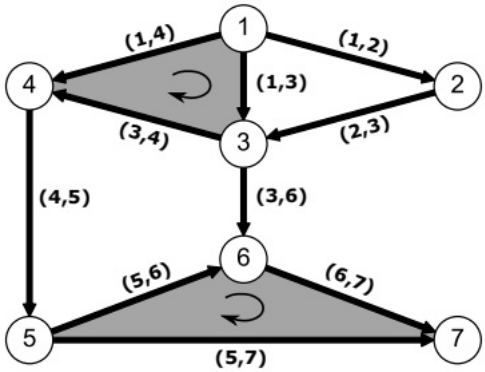

Figure 10: An example of a simplicial complex (Schaub et al., 2021). The 2-simplex $(5,6,7)$ is a co-face of the 1-simplices $(5,6)$,$(5,7)$,$(6,7)$. An example of a simplicial complex is then $S = \{(5,6,7),(5,6),(5,7),(6,7),(5),(6),(7)\}$

For computational purposes simplices are endowed with an orientation. Usually this is done by fixing an ordering of the vertices (e.g. $\{1,2,3,4\}$) which then induces an orientation. As an example consider Figure 10. Here the direction of all edges (1-simplices)

corresponds to the ordering of the nodes e.g. $(1 \to 4)$, $(5 \to 7)$, $(4 \to 5)$. Moreover, we can observe a signal on top of a simplicial complex. This is formally defined as:

Definition 7.2 (simplicial signal)**.** A $p-$simplicial signal x^p is a mapping from a $p-$simplex to the set of real numbers $\mathbb{R}^{N_p}$, where N_p is the total number of $p-$simplices. This can be formalized in terms of a vector $\mathbf{x}^p = [x_1^p, \ldots, x_{N_p}^p]^\top$, with x_i^p being the signal on the ith simplex.

An edge flow is denoted as $\mathbf{x}^1 = [x_1^1, \ldots, x_{N_1}^1]^\top$ where x_e^1 is the flow on the edge $e = (m, n)$ in $\mathcal{S}^1$. The experiments in this thesis focus on edge flows because of their wide applicability and for ease of exposition we drop the superscript and denote them as $\mathbf{x}$.

The relationship structure of a simplicial complex can be mathematically represented via incidence matrices $\mathbf{B}_p$ that encode the relationships between the $p-1$ and p simplices (with $\mathbf{B}_0 = 0$). The entries of the edge to node incidence matrix $\mathbf{B}_1$ are defined as

$$\mathbf{B}_{ie} = \begin{cases} 1 & \text{e = [j,i]} \\ -1 & \text{e = [i,j]} \\ 0 & \text{otherwise} \end{cases} . \tag{31}$$

The incident matrices for the other dimensions can be defined similarly, and this gives the following matrices for the simplices defined in Figure 4:

$$\mathbf{B}_1 = \begin{array}{c} \begin{matrix} (1,2) & (1,3) & (1,4) & (2,3) & (3,4) & (3,6) & (4,5) & (5,6) & (5,6) & (6,7) \end{matrix} \\ \begin{pmatrix} -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} \end{array} \tag{32}$$

$$\mathbf{B}_2 = \begin{array}{c} \begin{matrix} (1,3,4) & (5,6,7) \end{matrix} \\ \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & -1 \\ 0 & 1 \end{pmatrix} \end{array} \begin{matrix} (1,2) \\ (1,3) \\ (1,4) \\ (2,3) \\ (3,4) \\ (3,6) \\ (4,5) \\ (5,6) \\ (5,7) \\ (6,7) \end{matrix} . \tag{33}$$

Based on these incidence matrices we can define higher-order analogs of the graph Laplacian that are called the Hodge-Laplacians. The general case is defined as $\mathbf{L}_p = \mathbf{B}_p^\top \mathbf{B}_p + \mathbf{B}_{p+1}\mathbf{B}_{p+1}^\top$ and for simplicial complexes of order 2 we have:

$$\begin{aligned} \mathbf{L}_0 &= \mathbf{B}_1\mathbf{B}_1^\top, \\ \mathbf{L}_1 &= \mathbf{L}_{1,\ell} + \mathbf{L}_{1,u} := \mathbf{B}_1^\top\mathbf{B}_1 + \mathbf{B}_2\mathbf{B}_2^\top, \\ \mathbf{L}_2 &= \mathbf{B}_2^\top\mathbf{B}_2 \end{aligned} \tag{2}$$

where $\mathbf{L}_0$, $\mathbf{L}_1$, $\mathbf{L}_2$ encode the neighborhood relationships between nodes, edges, and triangles, respectively. Note, that $\mathbf{L}_0$ coincides with the graph Laplacian. Furthermore, $\mathbf{L}_{1,u} = \mathbf{B}_2\mathbf{B}_2^\top$ is called the upper-Laplacian and it encodes edge adjacencies that are due to common triangles, while the lower Laplacian $\mathbf{L}_{1,\ell} = \mathbf{B}_1^\top\mathbf{B}_1$ contains information about the relations due to common vertices.

7.1.2 The Hodge Decomposition on Edge Flows

The Hodge Decomposition is a natural decomposition of the signal space $\mathbb{R}^{N_p}$ into three orthogonal subspaces:

$$\mathbb{R}^{N_p} = \text{im}\left(\mathbf{B}_{p+1}\right) \oplus \text{im}\left(\mathbf{B}_p^\top\right) \oplus \ker\left(\mathbf{L}_p\right) \tag{34}$$

where $\text{im}(\cdot)$ and $\ker(\cdot)$ are the image and kernels spaces of a matrix and $\oplus$ is the direct sum of vector spaces. Thus, we can decompose any data on top of the complex, such as an edge flow, into three parts each of them living in one of three orthogonal subspaces $\mathbf{x} = \mathbf{x}_\text{G} + \mathbf{x}_\text{C} + \mathbf{x}_\text{H}$. In the case of an edge flow each of these three components is a flow

with a specific interpretable property. The flow $\mathbf{x}_G \in \text{im}\left(\mathbf{B}_1^\top\right)$ is the gradient flow. Its identifying property is that the sum of the flows going in and out of a node equals zero. The flow $\mathbf{x}_C \in \text{im}\left(\mathbf{B}_2\right)$ is the curl flow and it contains components of local circulations. For a simplicial complex that could be flows that go around triangles. The kernel space of the Laplacians, $\ker\left(\mathbf{L}_1\right)$ is the harmonic space and the flows $\mathbf{x}_H \in \ker\left(\mathbf{L}_1\right)$ represent all global cyclic flows that can not be represented in terms of local curls. Figure 11 depicts an example of such a decomposition.

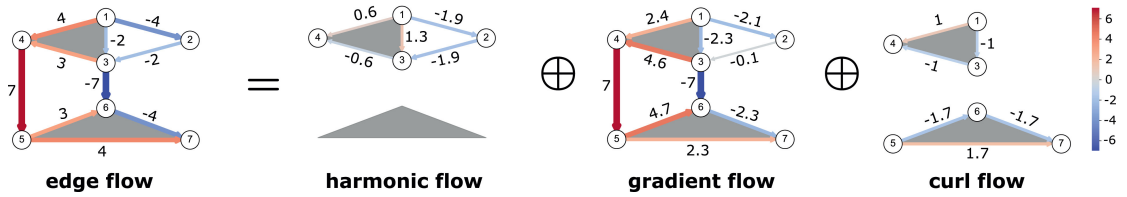

Figure 11: The Hodge Decomposition of a flow on the edges of a simplicial complex (Schaub et al., 2021).

Importantly, the eigenvectors of $\mathbf{L}_1$ span each of these three subspaces. As Hodge Laplacians are positive semi-definite, $\mathbf{L}_1$ has non-negative eigenvalues which can be interpreted as (non-negative) frequencies (sorted eigenvalues). The eigenvectors associated with the zero eigenvalues span the harmonic space, the eigenvectors corresponding to the nonzero eigenvalues of the upper laplacians $\mathbf{L}_{1,u}$ span the curl space and the nonzero eigenvalues of lower laplacian $\mathbf{L}_{1,l}$ span the gradient space. We can thus decompose the sets of eigenvalues and eigenvectors of $\mathbf{L}_1$ as $\mathbf{\Lambda}_1 = [\mathbf{\Lambda}_H\ \mathbf{\Lambda}_G\ \mathbf{\Lambda}_C]$ and $\mathbf{U}_1 = [\mathbf{U}_H\ \mathbf{U}_G\ \mathbf{U}_C]$ respectively, with the subscripts denoting harmonic, gradient and curl components. A flow $\mathbf{x}$ can then be projected onto the spaces spanned by the eigenvectors. This gives rise to the following embeddings:

$$\begin{aligned}
\tilde{\mathbf{x}}_H &= \mathbf{U}_H^\top \mathbf{x} = \mathbf{U}_H^\top \mathbf{x}_H \in \mathbb{R}^{N_H}, \text{ harmonic embedding} \\
\tilde{\mathbf{x}}_G &= \mathbf{U}_G^\top \mathbf{x} = \mathbf{U}_G^\top \mathbf{x}_G \in \mathbb{R}^{N_G}, \text{ gradient embedding} \\
\tilde{\mathbf{x}}_C &= \mathbf{U}_C^\top \mathbf{x} = \mathbf{U}_C^\top \mathbf{x}_C \in \mathbb{R}^{N_C}, \text{ curl embedding.}
\end{aligned}$$

7.1.3 Filters on Simplicial Complexes

We can use the Hodge Laplacian to build filters and define a Fourier transform on more general simplicial Complexes. In the same way as we have done with the shift operator on Graphs we can eigendecompose the positive semi-definite Hodge Laplacian:

$$\mathbf{L}_p = \mathbf{U}_p \mathbf{\Lambda}_p \mathbf{U}_p^\top = \sum_k^{N_p} \lambda_{p,k} \mathbf{u}_{p,k} \mathbf{u}_{p,k}^\top \tag{35}$$

with $\mathbf{U}_p = \left[\mathbf{u}_{p,1}, \ldots, \mathbf{u}_{p,N_p}\right]$ being the normalised eigenvectors and $\mathbf{\Lambda}_p = \text{diag}\left(\lambda_{p,1}, \ldots, \lambda_{p,N_p}\right)$ the associated eigenvalues. As this thesis considers the edge space we will drop the subscript p in what is to follow. When not specified differently we let $\mathbf{U} = \mathbf{U}_1$, $\lambda_k = \lambda_{1,k}$, $\mathbf{u}_k = \mathbf{u}_{1,k}$ etc.. With respect to the Hodge Laplacian as a reference operator we can then define a simplicial filter. A simplicial filter is an operator that acts separately on each of the eigenspaces depending on its eigenvalue. That is any function:

$$\begin{aligned} h : \mathbb{C} &\to \mathbb{R} \\ \lambda &\to h(\lambda) \end{aligned} \tag{36}$$

defines a filter

$$\mathbf{H} = \sum_k h\left(\lambda_k\right) \mathbf{u}_k \mathbf{u}_k^\top. \tag{37}$$

To illustrate such a filter, consider a signal $\mathbf{x}$ written in terms of the eigenbasis i.e. $\mathbf{x} = \sum_k \alpha_k \mathbf{u}_k$. Then we have:

$$\mathbf{y} = \mathbf{H}\mathbf{x} = \sum_k h\left(\lambda_k\right) \alpha_k \mathbf{u}_k \tag{38}$$

In that sense a filter, attenuates or increases the importance of the components of the signal depending on the eigenspace they lie in. The frequency response function that controls how this is done is $h\left(\lambda_k\right)$. To see how this can be used in practice, consider the Finite Impulse Response (FIR) Filter for edge flows introduced by Yang et al. (2021):

$$\mathbf{H}^{FIR} = \epsilon \mathbf{I} + \sum_{l_1=0}^{L_1} \alpha_{l_1} (\mathbf{L}_{1,l})^{l_1} + \sum_{l_2=0}^{L_2} \beta_{l_2} (\mathbf{L}_{1,u})^{l_2} \tag{39}$$

where ϵ, $\alpha = \{\alpha_1, ..., \alpha_{L_1}\}$ and $\beta = \{\beta_1, ..., \beta_{L_2}\}$ are the filter coefficients and L_1, L_2 specify the order of the filter. The frequency response (appendix F) is:

$$h\left(\lambda_i\right) = \begin{cases} \epsilon, \text{ for } \lambda_i \in \mathbf{\Lambda}_\text{H} \\ \epsilon + \sum_{l_1=1}^{L_1} \alpha_{l_1} \lambda_i^{l_1}, \text{ for } \lambda_i \in \mathbf{\Lambda}_\text{G} \\ \epsilon + \sum_{l_2=1}^{L_2} \beta_{l_2} \lambda_i^{l_2}, \text{ for } \lambda_i \in \mathbf{\Lambda}_\text{C} \end{cases} \tag{40}$$

By applying this FIR filter to a signal, we can modulate the different frequency components related to the Hodge decomposition. The parts of the signal that lie on each of these components can have a physical meaning and removing or increasing the strength of them is often beneficial to solve problems in real-world applications.

7.1.4 Simplicial Neural Networks

The FIR filter has a number of useful properties for processing signals on edge flows. As we have seen, it can accurately manipulate the Hodge information that is contained in the data. Moreover, it has a constant number of parameters and linear computational complexity in the number of edges (Yang et al., 2021). To also learn non-linear mappings, we can stack a number of these filters and combine them with pointwise non-linearities to obtain a simplicial convolutional neural network (SCNN) (Yang et al., 2022). Specifically, given the SCNN input $\mathbf{x}_0 := \mathbf{x}$, the propagation rule at each layer t is:

$$\mathbf{x}_t = \sigma\big(\mathbf{H}_t^{FIR}\mathbf{x}_{t-1}\big) \tag{41}$$

where $\sigma(\cdot)$ is the pointwise nonlinearity (e.g. ReLU).

7.2 The InfoMin Principle for Augmentations

In this section, we will discuss the InfoMin principle, a common principle based on which augmentations are designed. Later on we will use it to design augmentations for simplicial complexes.

Suppose that we generate two augmentations (or views) $\mathbf{x}'_1$,$\mathbf{x}'_2$ from a data point $\mathbf{x}$. Then Tian et al. (2020) argue that the optimal augmentation depends on the downstream task and can be theoretically reasoned about by considering the mutual Information of $\mathbf{x}$ and its label $\mathbf{y}$. There then exist the following alternative scenarios:

- $I(\mathbf{x}'_1;\mathbf{x}'_2) < I(\mathbf{x};\mathbf{y})$ performance is degraded because task-relevant information is discarded in the views.
- $I(\mathbf{x}'_1;\mathbf{y}) = I(\mathbf{x}'_2;\mathbf{y}) = I(\mathbf{x}'_1;\mathbf{x}'_2) = I(\mathbf{x};\mathbf{y})$ only task-relevant information is shared between the views and there is no irrelevant noise.
- $I(\mathbf{x}'_1;\mathbf{x}'_2) > I(\mathbf{x};\mathbf{y})$ irrelevant information is included which potentially leads to worse generalisation on the downstream task.

From this the InfoMin principle is derived which states that: ”A good set of views are those that share the minimal information necessary to perform well at the downstream task”.

It complements the InfoMax principle (the mutual information should be maximized) by arguing that maximizing mutual information is only useful as long as it is task-relevant. In general, it is impossible for us to know what exactly the task-relevant information is. Nevertheless, we can try to approximate it by making use of our prior knowledge about the task and design appropiate augmentations based on that.

8 Spectral Methods for Contrastive Learning on Simplicial Complexes

Simplicial complexes offer a powerful inductive bias for edge flow data. Importantly, their spectrum can be decomposed into three interpretable subspaces via the Hodge decomposition capturing the gradient, curl and harmonic parts of a flow. Capitalizing on this spectral decomposition, we introduce a contrastive self-supervised learning methodology to process simplicial data. Building up upon that we design an augmentation method that generates positive contrastive examples with desirable spectral characteristics encouraging the network to encode Hodge information into the embeddings. Additionally, we reweight the negative examples in the contrastive loss based on the affinity of their Hodge components to the anchor. This amplifies the separation between spectrally dissimilar instances. We empirically validate the approach on two trajectory prediction tasks. This chapter is based on the recent paper *Hodge-aware Contrastive Learning*, which is currently under review at ICASSP (Möllers et al., 2023).

8.1 Contrastive Learning on Simplicial Complexes (SCL)

To extend self-supervised learning to be able to process data on edges we make use of the general contrastive learning framework introduced by (Chen et al., 2020) and adjust the encoder and augmentations (Figure 12). Applied to an edge flow $\mathbf{x}$, we create both positive and negative examples and train an SCNN as an encoder. As a loss function, the temperature-scaled InfoNCE is used as introduced in eq. (19) and eq. (21). Negative samples are taken to be all other data points from the same batch. For the experiments, we take an augmentation method from the graph domain (Node Feature Masking) and implement what can be considered its counterpart on edge flows.

Node Feature Masking $\rightarrow$Edge Flow Masking

Instead of masking node features on graphs, the edge features of a simplicial complex are masked. This is implemented by sampling a mask from a uniform Bernoulli distribution. Edge flow features are masked with probability p to generate a positive example. This process can be written as $\mathbf{x}' = \mathcal{T}(\mathbf{x}) := \mathbf{x} \circ \mathbf{e}$ where $\mathbf{e}$ is a random Bernoulli vector with entry $\mathbf{e}_i \sim Ber(\mathbf{p}_i)$ and $\circ$ is the elementwise product. Usually the masking probability for all edges is the same.

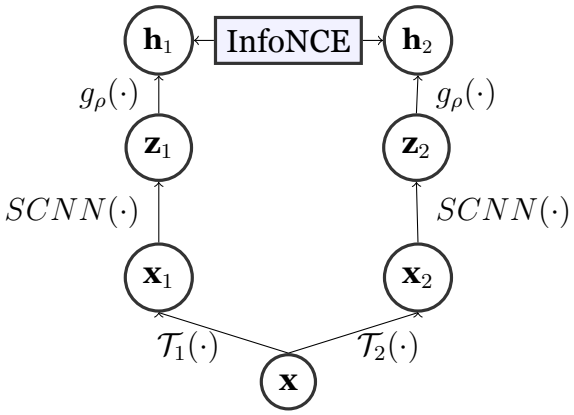

Figure 12: In Contrastive learning for simplicial complexes we augment the data point $\mathbf{x}$ (anchor) with two transformations $\mathcal{T}_{1\backslash 2}(\cdot)$ to generate positive examples. The latter are then passed through an SCNN encoder to generate the simplicial embeddings $\mathbf{z}$. After that a projection head $g_\rho(\cdot)$ is applied and the InfoNCE loss is computed. The simplicial structure equips the encoder with a useful inductive bias to process edge flows.

8.2 Spectral Augmentations for Edge Flows

The task-relevant information on problems that are solved on simplices is different than the one that is required for problems on graphs. One example are edge flow prediction problems where the components of the Hodge Decomposition are inherently relevant. In these problem settings we want to design simplicial augmentations that destroy information on irrelevant Hodge embeddings and preserve it on the others (InfoMin principle). In the following we present an optimization-based approach to tackle this problem, a less effective alternative, relying on spectral filters, is deferred to Appendix G.

8.2.1 Spectrally-Optimized Dropout

We now show how the Edge Flow Masking augmentation can be made hodge-aware. To do this, we formalize an optimisation problem that helps us find dropout probabilities $\mathbf{p}$. We optimize the expected value of the difference of the generated Hodge embeddings to the embeddings of the anchor. More specifically, denote the Hodge embeddings generated via an Edge Flow Masking augmentation as $\tilde{\mathbf{x}}'_{\mathrm{G}} = \mathbf{U}_{\mathrm{G}}^\top \mathbf{x}'$, $\tilde{\mathbf{x}}'_{\mathrm{C}} = \mathbf{U}_{\mathrm{C}}^\top \mathbf{x}'$, $\tilde{\mathbf{x}}'_{\mathrm{H}} = \mathbf{U}_{\mathrm{H}}^\top \mathbf{x}'$. Then the expected quadratic differences $\mathcal{L}_{\mathrm{G}}(\mathbf{p}) = \mathbb{E}[\|\tilde{\mathbf{x}}_{\mathrm{G}} - \tilde{\mathbf{x}}'_{\mathrm{G}}\|_2^2]$, $\mathcal{L}_{\mathrm{C}}(\mathbf{p}) = \mathbb{E}[\|\tilde{\mathbf{x}}_{\mathrm{C}} - \tilde{\mathbf{x}}'_{\mathrm{C}}\|_2^2]$, $\mathcal{L}_{\mathrm{H}}(\mathbf{p}) = \mathbb{E}[\|\tilde{\mathbf{x}}_{\mathrm{H}} - \tilde{\mathbf{x}}'_{\mathrm{H}}\|_2^2]$ can the be used to quantify how different each hodge component of the augmentation is from the anchor. We can expand the expression for the gradient embedding (the same holds for the others):

$$\mathbb{E}[\|\tilde{\mathbf{x}}_{\mathrm{G}} - \tilde{\mathbf{x}}'_{\mathrm{G}}\|_2^2] = \|\tilde{\mathbf{x}}_{\mathrm{G}}\|_2^2 - \mathrm{Tr}\left(\mathbf{U}_{\mathrm{G}}\mathbf{x}(\mathbf{x}\circ\mathbf{p})^\top \mathbf{U}_{\mathrm{G}}^\top\right) - \mathrm{Tr}\left(\mathbf{U}_{\mathrm{G}}(\mathbf{x}\circ\mathbf{p})\mathbf{x}^\top \mathbf{U}_{\mathrm{G}}^\top\right) + \mathrm{Tr}\left(\mathbf{U}_{\mathrm{G}}\mathbf{P}\mathbf{U}_{\mathrm{G}}^\top\right) \quad (42)$$

where $\mathbf{P}$ is a matrix with entries $\mathbf{P}_{i,j} = \mathbf{x}_i\mathbf{x}_j\mathbf{p}_i\mathbf{p}_j$ for $i \neq j$ and $\mathbf{P}_{i,j} = (\mathbf{x}_i)^2\mathbf{p}_i$ for $i = j$. To obtain the above expression, it is important to recall the equality $\text{Tr}\left(\mathbf{X}\mathbf{X}^\top\right) = \|\mathbf{X}\|_2^2$ and that $\mathbb{E}\left[\mathbf{x} \circ \mathbf{e}\right] = \mathbf{x} \circ \mathbf{p}$. Based on this we can optimize $\mathbf{p}$ such that the distance to the anchor for one or more of the augmented embeddings is small/large. For example, in applications in which the the curl and harmonic embeddings are important we solve the following problem to obtain $\mathbf{p}$:

$$\min_{\mathbf{p}} \quad -\mathcal{L}_{\text{G}}(\mathbf{p}) + \mathcal{L}_{\text{C}}(\mathbf{p}) + \mathcal{L}_{\text{H}}(\mathbf{p}) \tag{43a}$$

$$\text{subject to} \quad \mathbf{p} \in \mathcal{G}_{\mathbf{p}} := \{\mathbf{p} \mid \mathbf{p} \in [0,1], \|\mathbf{p}\|_1 \leq \epsilon_{\mathbf{p}}\}\,, \tag{43b}$$

where the set $\mathcal{G}_{\mathbf{p}}$ puts a maximum budget $\epsilon_{\mathbf{p}}$ on the masked edge values. By solving this we obtain dropout probabilities $\mathbf{p}$ under which positive data points with similar curl and harmonic components to the anchor are generated but with a different gradient component (in expectation). To obtain $\mathbf{p}$ we us a projected gradient descent algorithm, projecting the solution onto the constraint set $\mathcal{G}_{\mathbf{p}}$ after every step.

8.3 Hodge Aware Debiasing

In the previous section we designed augmentations to introduce knowledge about the Hodge Decomposition into the embedding space. To further encourage a spectral organization of the latter, we shall also act on the negative samples. This is known as a *debiasing* technique and consists of reweighting the denominator in the InfoNCE loss:

$$\mathcal{L} = -\mathop{\mathbb{E}}_{X}\left[\log \frac{f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right)}{\sum_{j=1}^{M} a(\mathbf{x}, \mathbf{x}_j) f_{\boldsymbol{\rho}}\left(\mathbf{x}_j, \mathbf{z}'\right)}\right] \tag{44}$$

where $a(\mathbf{x}, \mathbf{x'}_j)$ is a weighting term between the anchor $\mathbf{x}$ and the negative example $\mathbf{x_j}$. To encourage a spectrally-organized embedding space these weights can be picked such that spectrally different samples are pushed further away from the anchor. To do this we introduce the weighted embedding similarity between two data points:

$$\mathcal{S}(\mathbf{x}, \mathbf{x}') = \gamma_{\text{H}}\,\text{CD}(\tilde{\mathbf{x}}_{\text{H}}, \tilde{\mathbf{x}}'_{\text{H}}) + \gamma_{\text{G}}\,\text{CD}(\tilde{\mathbf{x}}_{\text{G}}, \tilde{\mathbf{x}}'_{\text{G}}) + \gamma_{\text{C}}\,\text{CD}(\tilde{\mathbf{x}}_{\text{C}}, \tilde{\mathbf{x}}'_{\text{C}}) \tag{45}$$

with $\text{CD}(\mathbf{x}, \mathbf{x}') = 1 - \frac{\mathbf{x}^\top \mathbf{x}'}{\|\mathbf{x}\|_2 \|\mathbf{x}'\|_2}$ being the cosine distance and $\gamma_{\text{H}}, \gamma_{\text{G}}, \gamma_{\text{C}} \geq 0$ hyperparameters. Importantely, the cosine distance assigns higher values to spectrally more dissimilar examples. Lastly, to ensures that the loss terms for different data points are comparable we normalize the similarity over the M negative samples and obtain a weight:

$$a(\mathbf{x}, \mathbf{x}_j) = \frac{\mathcal{S}(\mathbf{x}_i, \mathbf{x'}_j)}{\sum_{j=1}^{N} \mathcal{S}(\mathbf{x}_i, \mathbf{x'}_j)} \tag{46}$$

Substituting (46) into (44) leads to a spectrally-reweighted contrastive loss. The hyperparameters can be used to control the specific notion of similarity. In the experiments in the next section, it is known that data points with different harmonic embeddings should be pushed away from each other. Therefore, we choose to calculate the weights based on the latter and pick $\gamma_G = 0$, $\gamma_C = 0$, $\gamma_H = 1$.

9 Experiments

9.1 Set-up

Datasets We evaluate all proposed methods on two edge flow classification tasks where the flows are represented as signals on an oriented SC. The first is a synthetic dataset consisting of trajectories that either pass through the bottom-right or top-left corner of a map and the task is to distinguish these two classes (Figure 13). The second is a real-world equivalent of this synthetic benchmark that contains ocean drifters moving around the island of Madagascar between years 2011-2018. Here we attempt to differentiate between clockwise and anti-clockwise moving drifters. A property of both datasets is that the information on the harmonic frequencies is important for classification due to holes in the underlying Simplicial Complex. For the trajectory dataset we generate 200 training, 100 validation and 100 test data points while there are 160 training examples and 40 test data points for the ocean drifter task. In addition to the result presented here we have also run experiments on lifted graphs. These can be found in Appendix H.

Experimental Procedure To evaluate the methods on the datasets described in the previous section we follow a standard transductive set-up introduced by (Zhu, Xu, Liu and Wu, 2021). That is, we train the Simplicial self-supervised learner on all available data points and afterwards train a linear support vector machine (SVM) on the generated embeddings. For the ocean drifter task no validation set is available and a 10-fold cross-validation is performed to estimate the hyperparamers. We evaluate the method on 16 data splits and report the mean accuracy.

Implementation & Training Details For the network architecture for the SCNN we follow the settings from Bodnar et al. (2021) (who propose a supervised model) and use a hidden-layer of size 64 and Tanh-activations. Furthermore, we optimize the number of encoder layers under the edge drop augmentation, pick between $[1, 2, 3]$-number of layers and add a projection head consisting of two-layers as in Chen et al. (2020). All parameters are optimized with stochastic gradient descent using the Adam algorithm (Kingma and Ba, 2015) and we grid search the learning rate and weight decay in the interval $[10^{-5}, 1]$ in decimal steps. The augmentation probabilities for the edge drop augmentation $\mathbf{p}$ and perturbation budged $\epsilon_{\mathbf{p}}$ are chosen in the interval $[0.1, 0.4]$ while for the Hodge Drop augmentation the interval $[0.1, 0.7]$ is searched.

Baselines As baselines we employ a fully-supervised SCNN for the edge flow classification tasks. Furthermore, we train a network that uses only lower Laplacian encoding (SCL_{low}). By omitting the triangle relationships we are able to quantify the impact of the Simplicial approach that includes these upper neighbourhoods. We complement all that with a sensitivity analysis for different augmentations and parameters.

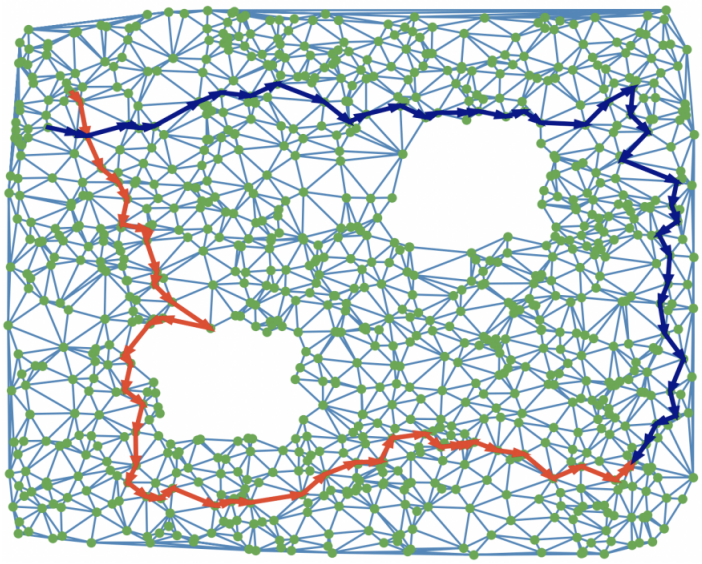

Figure 13: Trajectories as Edge Flows on a Simplicial Complex. The structure of the Simplicial Complex represents a map with two holes (e.g. islands). The different cyclic movement of the two displayed trajectories around these holes leads them to have different harmonic components that can be used to distinguish them (Bodnar et al., 2021).

9.2 Results

The first observation that we make in our experiments is that the promising performance of self-supervised learning extends to edge flows (Table 2). Notably, our proposed method ($SSCL_{Spec}$), that uses a reweighted loss function and spectrally optimized Edge Flow Masking augmentations, achieves the best accuracy on both trajectory datasets. This experimentally validates that our approach encourages the encoding of (harmonic) hodge information into the embeddings. This aspect is underscored by Figure 14 that shows the embedding distance for two different augmentation techniques (uniform edge feature masking and proposed). The proposed approach generates more similar harmonic embeddings, which is key to the obtained results for the task. Moreover, spectrally reweighing the loss functions improves the performance of the learner independent of the augmentation quality (Figure 15).

Table 2: Test Accuracies for the Trajectory and Ocean Drifter datasets. SCL denotes models trained with the standard InfoNCE loss, while SSCL models are trained with spectrally reweighted negatives from section 8.3. The subscript Spec denotes that the augmentation probablities are spectrally optimized.

Model	Trajectory Task	Ocean Drifters
$\text{SSCL}_{\text{Spec}}$ (ours)	97.9 ± 0.3	90.3 ± 1.4
SCNN (supervised)	95.2 ± 0.5	78.5 ± 1.1
SSCL	96.8 ± 0.4	89.1 ± 1.0
SCL_{Spec}	98.2 ± 0.4	83.1 ± 1.1
SCL	96.1 ± 0.6	81.6 ± 1.6
SCL_{low}	91.0 ± 0.2	77.1 ± 1.2

Notably, even for a model that uses the standard InfoNCE loss, the spectrally optimized augmentations (SCL_{Spec}) yield better downstream accuracy than then ones with uniform probabilities (SCL). This emphasizes that the spectral augmentations are a standalone feature that is also useful independent of the loss function. Furthermore, to evaluate the impact of the simplicial structure we tested a learner that uses only lower Laplacian connections (SCL_{low}), omitting triangle relationships. Compared to its simplicial counterpart SCL under identical conditions, SCL_{low}, manifests a noticeable decrease in performance. This demonstrates that the structural advantages of simplicial networks to process flow data transfer to the contrastive learning setting.

10 Discussion

We have shown that a simplicial contrastive learning framework is an effective method for generating representations of edge flows that contain hodge information. Two important factors for this are the useful inductive bias that the simplicial structure provides for flow data and augmentations that respect spectral information. We designed the latter by optimizing the probabilities of a dropout augmentation so that it preserves the harmonic information in the data. One important aspect of this approach is that the augmented examples have spectrally desirable properties while being topological similar to the anchor data point due to the dropout mechanism used. In that sense our approach considers both the spectral and the spatial domain. Moreover, we introduced a weighting term into the loss function that pushes spectrally different examples further apart and encourages an

organisation of the embedding space that takes the hodge information into account. This weighting is particularly beneficial in settings in which prior knowledge exists that dictates which hodge information should be encoded into the embeddings. In these settings the spectrally optimized loss shines and we can generate good embeddings even without optimal augmentations. Our empirical evaluations validate the utility of our approaches and a classifier using the optimized representations can outperform fully-supervised models in edge flow classification tasks.

11 Related Work

To the best of our knowledge this is the first work that proposes contrastive learning for simplicial complexes. It builds upon a flurry of recent advances that have enabled and popularized the use of SCs for data processing (Battiston et al., 2020). Only recently, a simplicial Fourier transform (Barbarossa and Sardellitti, 2020) and convolutional filters (Isufi and Yang, 2022) have been developed and applied to neural networks (Ebli et al., 2020, Roddenberry et al., 2021, Bodnar et al., 2021). These methods have proven particularly effective when applied to the processing of flow data (Barbarossa and Sardellitti, 2020, Roddenberry et al., 2022). Notably, Krishnan et al. (2023) use them to alleviate the curse of dimensionality in autoregressive flow prediction for water networks and Isufi and Yang (2022) leverage simplicial filters to remove arbitrage opportunities in currency exchange markets.

The idea to optimize the dropout probabilities in contrastive learning to generate better examples is inspired by related work on graphs. Zhu, Xu, Yu, Liu, Wu and Wang (2021) drop nodes with high node centrality less frequently and mask task-irrelevant features more often, while Liu et al. (2022) adjust augmentations on homophilic graphs such that the change in high frequency information is less than the change in the more important low frequency domain. Moreover, Lin et al. (2023) pick dropout probabilities such that they maximize the spectral difference between the eigenvalues of augmented Laplacians and show that this leads to an improved performance on downstream tasks.

The spectrally reweighted InfoNCE loss is related to the idea of *debiasing* in contrastive learning. Debiasing methods reweigh the terms in the loss in order to reduce the impact of false negatives in the denominator (Chuang et al., 2020, Sun et al., 2023, Liu et al., 2023). To the best of our knowledge, no existing work uses this approach to encourage a spectral organization of the embedding space.

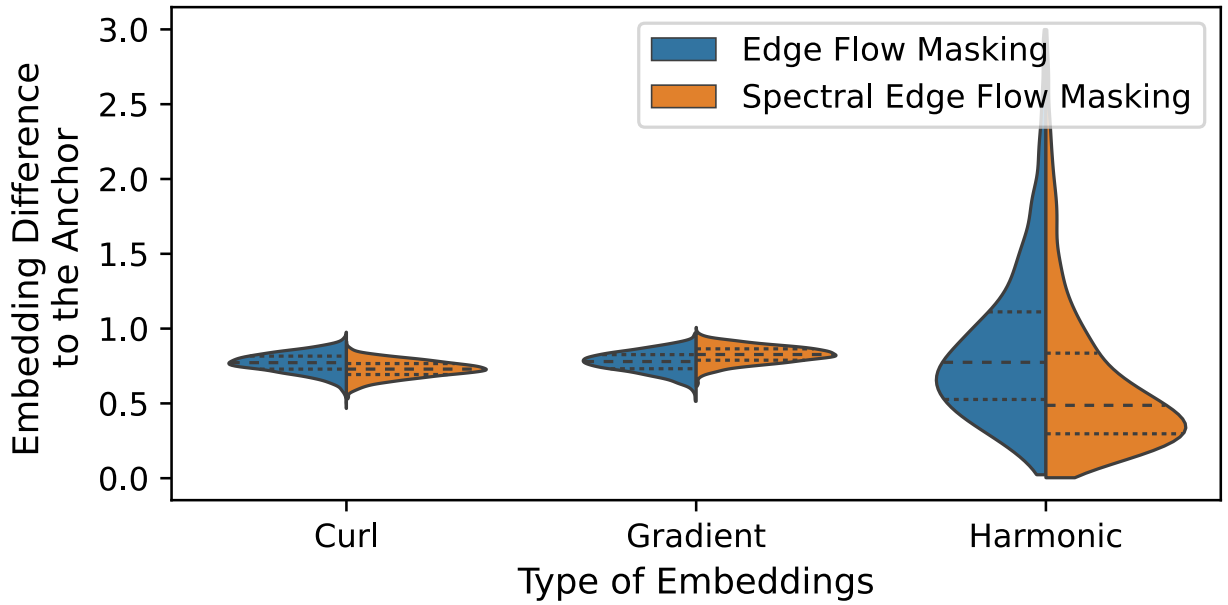

Figure 14: This plot is generated by augmenting data points by masking edge features. The distribution of the difference between the hodge embeddings of the augmented examples and the anchor is plotted. For an augmentation with spectrally optimized probabilities (orange) more probability mass lies over smaller differences in the harmonic embedding and we are thus more likely to generate samples with more similar harmonic components to the anchor than when using uniform probabilities (blue).

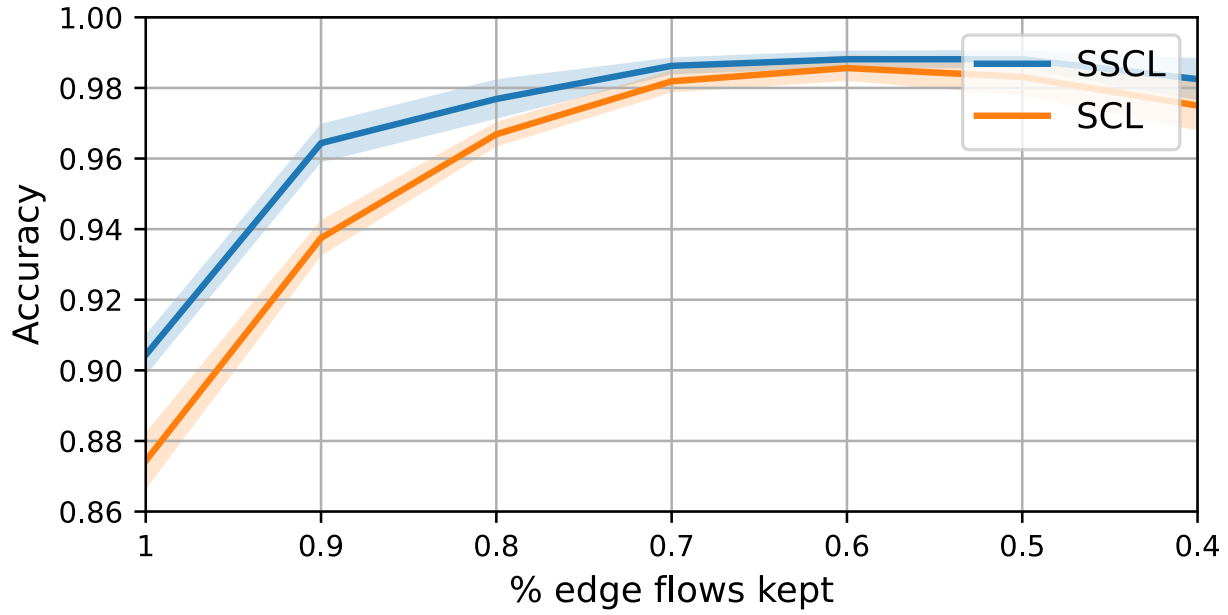

Figure 15: Comparison of downstream accuracy between a model trained with a spectrally reweighted loss (SSCL) and one with trained with the standard loss (SCL) for an Edge Feature Masking augmentation. The SSCL outperforms the SCL irrespective of the augmentation quality.

12 Conclusion

This thesis tackled two underexplored areas in the CL literature. It set out to (1) establish a notion of uncertainty in contrastive learning and to (2) investigate how it can be extended to simplicial complexes in order to generate useful representations for edge flow data. We addressed the first challenge by leveraging a probabilistic interpretation of the InfoNCE loss to train a (Bayesian) variational graph contrastive learner (VGCL). This equips the model with a notion of epistemic uncertainty over the parameters. Building up upon that we developed the contrastive model disagreement score (CMDS) that combines the epistemic uncertainty over the weights with the disagreement in likelihoods for different positive examples. We empirically evaluated the CMDS on standard node classification tasks and showed that it leads to better calibrated uncertainties than existing methods. Particularly, it is a powerful indicator for downstream performance and could find applications in active learning or out-of-distribution detection. In the second part of this thesis we then developed a simplicial approach to contrastive learning. For this we employed a Simplicial Convolutional Neural Network as an encoder and used it to generate representations for edge flow data. Crucially, for flow data a lot of information lies on the components of the so-called Hodge decomposition of the spectrum. To incorporate this into our approach we designed a spectrally-optimized augmentation method that leaves relevant hodge information intact. Moreover, we reweighted the significance of negative examples in the contrastive loss, considering the similarity of their Hodge components to the anchor. Doing this encourages a stronger separation among less similar instances and it leads to an embedding space that reflects the spectral properties of the data. We tested the proposed approaches on two edge flow datasets and showed that they generate embeddings that contain more relevant hodge information, leading to an improved accuracy in downstream classification tasks.

Building up upon the methods developed in this thesis there are a number of avenues for future exploration. Firstly, we have not investigated different approximation methods for Bayesian Inference. Alternative methods such as Markov Chain Monte Carlo or Laplace approximations have the potential to yield superior performance or reduce the computational cost. The latter is of utmost importance in scenarios with vast and complex data, where training times can extend over several days (e.g. on the Imagenet benchmark). Related to this, another important point is the generalizability of the ideas proposed in this thesis. While the experiments are limited to topological structures, the introduced probabilistic methods are general and can easily be extended to other applications such as images or time series. This is especially interesting as the InfoNCE has already shown promising re-

sults in these domains. With respect to simplical contrastive learning, exploring additional methods for data augmentation and conducting experiments with simplicial complexes of varying dimensions remain as open research directions. Moreover, extending our optimization approach to these different types of augmentations beyond dropout would lead to a more diverse range of methods that can create positive examples with specific spectral properties.

In a broader context, our work has implications for the way we understand and develop contrastive learning. By successfully integrating a method for variational autoencoders into the CL framework, we have demonstrated that the interpretation as a probabilistic model allows for the transfer of established methods from the rich literature on generative models to contrastive learning. This cross-pollination between the two fields opens up a pathway for future exploration into how other probabilistic techniques can be ported to enhance self-supervised models. Considering the limited amount of existing work on uncertainty in CL and the growing popularity of these kinds of models, the importance of this is not to be underestimated. On the spectral side, our work accentuates the pivotal role that such a Hodge perspective can play, particularly when dealing with data on simplicial complexes. While much of the existing research in the CL literature has been predominantly focused on leveraging spatial features, our work serves as a critical reminder of the untapped potential residing in the spectral domain. Especially in the context of edge flows, the spectral space is rich with structured information and this thesis has demonstrated that incorporating it is indispensable for generating high-quality representations of the data.

References

Aitchison, L. and Ganev, S. (2023), 'Infonce is variational inference in a recognition parameterised model'.
URL: *https://arxiv.org/pdf/2107.02495.pdf*

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A. Q., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M. and Farhan, L. (2021), 'Review of deep learning: concepts, cnn architectures, challenges, applications, future directions', *Journal of Big Data* **8**.

Ardeshir, S. and Azizan, N. (2022), Embedding reliability: On the predictability of downstream performance, *in* 'NeurIPS ML Safety Workshop'.
URL: *https://openreview.net/forum?id=TedqYedIERd*

Balestriero, R., Ibrahim, M., Sobal, V., Morcos, A., Shekhar, S., Goldstein, T., Bordes, F., Bardes, A., Mialon, G., Tian, Y., Schwarzschild, A., Wilson, A. G., Geiping, J., Garrido, Q., Fernandez, P., Bar, A., Pirsiavash, H., LeCun, Y. and Goldblum, M. (2023), 'A cookbook of self-supervised learning'.

Barbarossa, S. and Sardellitti, S. (2020), 'Topological signal processing over simplicial complexes', *IEEE Transactions on Signal Processing* **68**, 2992–3007.

Bardes, A., Ponce, J. and LeCun, Y. (2022), VICReg: Variance-invariance-covariance regularization for self-supervised learning, *in* 'International Conference on Learning Representations'.
URL: *https://openreview.net/forum?id=xm6YD62D1Ub*

Battiston, F., Cencetti, G., Iacopini, I., Latora, V., Lucas, M., Patania, A., Young, J.-G. and Petri, G. (2020), 'Networks beyond pairwise interactions: Structure and dynamics', *Physics Reports* **874**, 1–92.
URL: *https://www.sciencedirect.com/science/article/pii/S0370157320302489*

Blundell, C., Cornebise, J., Kavukcuoglu, K. and Wierstra, D. (2015), Weight uncertainty in neural networks, *in* 'ICML'.

Bodnar, C., Frasca, F., Wang, Y., Otter, N., Montufar, G. F., Lió, P. and Bronstein, M. (2021), Weisfeiler and Lehman go topological: Message passing simplicial networks, *in* M. Meila and T. Zhang, eds, 'Proceedings of the 38th International Conference on Machine Learning', Vol. 139 of *Proceedings of Machine Learning Research*, PMLR, pp. 1026–1037.

Brüel Gabrielsson, R. (2020), Universal function approximation on graphs, *in* H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin, eds, 'Advances in Neural Information Processing Systems', Vol. 33, Curran Associates, Inc., pp. 19762–19772.

Chen, T., Kornblith, S., Norouzi, M. and Hinton, G. (2020), A simple framework for contrastive learning of visual representations, *in* 'Proceedings of the 37th International Conference on Machine Learning', ICML'20.

Chuang, C.-Y., Robinson, J., Lin, Y.-C., Torralba, A. and Jegelka, S. (2020), Debiased contrastive learning, *in* H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin, eds, 'Advances in Neural Information Processing Systems', Vol. 33, Curran Associates, Inc., pp. 8765–8775.

Daxberger, E. A. and Hernández-Lobato, J. M. (2019), 'Bayesian variational autoencoders for unsupervised out-of-distribution detection'.
URL: *http://arxiv.org/abs/1912.05651*

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M. and Hennig, P. (2021), Laplace redux - effortless bayesian deep learning, *in* M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang and J. W. Vaughan, eds, 'Advances in Neural Information Processing Systems', Vol. 34, Curran Associates, Inc., pp. 20089–20103.

Ebli, S., Defferrard, M. and Spreemann, G. (2020), Simplicial neural networks, *in* 'TDA & Beyond'.
URL: *https://openreview.net/forum?id=nPCt39DVIfk*

Gal, Y. and Ghahramani, Z. (2016), Dropout as a bayesian approximation: Representing model uncertainty in deep learning, *in* M. F. Balcan and K. Q. Weinberger, eds, 'Proceedings of The 33rd International Conference on Machine Learning', Vol. 48 of *Proceedings of Machine Learning Research*, PMLR, New York, New York, USA, pp. 1050–1059.
URL: *https://proceedings.mlr.press/v48/gal16.html*

Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep Learning*, MIT Press. http://www.deeplearningbook.org.

Graves, A. (2011), Practical variational inference for neural networks, *in* J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira and K. Weinberger, eds, 'Advances in Neural Information Processing Systems', Vol. 24, Curran Associates, Inc.

Hasanzadeh, A., Armandpour, M., Hajiramezanali, E., Zhou, M., Duffield, N. and Narayanan, K. (2021), 'Bayesian graph contrastive learning'.
URL: *https://arxiv.org/abs/2112.07823*

He, K., Fan, H., Wu, Y., Xie, S. and Girshick, R. (2020), Momentum contrast for unsupervised visual representation learning, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 9729–9738.

Hornik, K., Stinchcombe, M. and White, H. (1989), 'Multilayer feedforward networks are universal approximators', *Neural Netw.* **2**(5), 359–366.

Isufi, E., Gama, F., Shuman, D. I. and Segarra, S. (2022), 'Graph filters for signal processing and machine learning on graphs', *arXiv preprint arXiv:2211.08854* .

Isufi, E. and Yang, M. (2022), Convolutional filtering in simplicial complexes, *in* 'ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', pp. 5578–5582.

Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D. and Makedon, F. (2021), 'A survey on contrastive self-supervised learning', *Technologies* **9**(1).
URL: *https://www.mdpi.com/2227-7080/9/1/2*

Jospin, L. V., Laga, H., Boussaid, F., Buntine, W. and Bennamoun, M. (2022), 'Hands-on bayesian neural networks—a tutorial for deep learning users', *IEEE Computational Intelligence Magazine* **17**(2), 29–48.

Kingma, D. and Ba, J. (2015), Adam: A method for stochastic optimization, *in* 'International Conference on Learning Representations (ICLR)', San Diega, CA, USA.

Kingma, D. P., Salimans, T. and Welling, M. (2015), Variational dropout and the local reparameterization trick, *in* C. Cortes, N. Lawrence, D. Lee, M. Sugiyama and R. Garnett, eds, 'Advances in Neural Information Processing Systems', Vol. 28, Curran Associates, Inc.

Kipf, T. N. and Welling, M. (2017), Semi-supervised classification with graph convolutional networks, *in* 'Proceedings of the 5th International Conference on Learning Representations', ICLR '17.
URL: *https://openreview.net/forum?id=SJU4ayYgl*

Krishnan, J., Money, R., Beferull-Lozano, B. and Isufi, E. (2023), Simplicial vector autoregressive model for streaming edge flows, *in* 'ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)'.

Li, Y., Pogodin, R., Sutherland, D. J. and Gretton, A. (2021), Self-supervised learning with kernel dependence maximization, *in* M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang and J. W. Vaughan, eds, 'Advances in Neural Information Processing Systems', Vol. 34, Curran Associates, Inc., pp. 15543–15556.

Lin, L., Chen, J. and Wang, H. (2023), Spectral augmentation for self-supervised learning on graphs, *in* 'The Eleventh International Conference on Learning Representations'.
URL: *https://openreview.net/forum?id=DjzBCrMBJ$_p$*

Liu, B. and Wang, B. (2023), 'Bayesian self-supervised contrastive learning'.

Liu, N., Wang, X., Bo, D., Shi, C. and Pei, J. (2022), Revisiting graph contrastive learning from the perspective of graph spectrum, *in* A. H. Oh, A. Agarwal, D. Belgrave and K. Cho, eds, 'Advances in Neural Information Processing Systems', Vol. 35.
URL: *https://openreview.net/forum?id=LoU7TUWRt$_X$*

Liu, Y., Yang, X., Zhou, S., Liu, X., Wang, Z., Liang, K., Tu, W., Li, L., Duan, J. and Chen, C. (2023), 'Hard sample aware network for contrastive deep graph clustering', *Proceedings of the AAAI Conference on Artificial Intelligence* **37**(7), 8914–8922.
URL: *https://ojs.aaai.org/index.php/AAAI/article/view/26071*

Lu, Z., Pu, H., Wang, F., Hu, Z. and Wang, L. (2017), The expressive power of neural networks: A view from the width, *in* I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds, 'Advances in Neural Information Processing Systems', Vol. 30, Curran Associates, Inc.

MacKay, D. J. (1992), 'A practical Bayesian framework for backpropagation networks', *Neural Computation* **4**(3).

Möllers, A., Immer, A., Isufi, E. and Fortuin, V. (2023), Uncertainty in graph contrastive learning with bayesian neural networks, *in* 'Fifth Symposium on Advances in Approximate Bayesian Inference'.
URL: *https://openreview.net/forum?id=LH76pl-OUj*

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P. and Neumann, M. (2020), Tudataset: A collection of benchmark datasets for learning with graphs, *in* 'ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)'.
URL: *www.graphlearning.io*

Möllers, A., Immer, A., Fortuin, V. and Isufi, E. (2023), 'Hodge-aware contrastive learning'.
URL: *https://arxiv.org/abs/2309.07364*

Neal, R. (1992), Bayesian learning via stochastic dynamics, *in* S. Hanson, J. Cowan and C. Giles, eds, 'Advances in Neural Information Processing Systems', Vol. 5, Morgan-Kaufmann.

Neal, R. M. et al. (2011), 'MCMC using Hamiltonian dynamics', *Handbook of Markov Chain Monte Carlo* **2**(11).

Poole, B., Ozair, S., Van Den Oord, A., Alemi, A. and Tucker, G. (2019), On variational bounds of mutual information, *in* K. Chaudhuri and R. Salakhutdinov, eds, 'Proceedings of the 36th International Conference on Machine Learning', Vol. 97 of *Proceedings of Machine Learning Research*, PMLR, pp. 5171–5180.
URL: *https://proceedings.mlr.press/v97/poole19a.html*

Robbins, H. E. (1951), 'A stochastic approximation method', *Annals of Mathematical Statistics* **22**, 400–407.

Roddenberry, T. M., Frantzen, F., Schaub, M. T. and Segarra, S. (2022), Hodgelets: Localized spectral representations of flows on simplicial complexes, *in* 'ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', pp. 5922–5926.

Roddenberry, T. M., Glaze, N. and Segarra, S. (2021), Principled simplicial neural networks for trajectory prediction, *in* M. Meila and T. Zhang, eds, 'Proceedings of the 38th International Conference on Machine Learning', Vol. 139 of *Proceedings of Machine Learning Research*, PMLR, pp. 9020–9029.
URL: *https://proceedings.mlr.press/v139/roddenberry21a.html*

Schaub, M. T., Zhu, Y., Seby, J.-B., Roddenberry, T. M. and Segarra, S. (2021), 'Signal processing on higher-order networks: Livin' on the edge... and beyond', *Signal Processing* **187**, 108149.

Sharma, M., Rainforth, T., Teh, Y. W. and Fortuin, V. (2023), 'Incorporating unlabelled data into bayesian neural networks', *arXiv preprint arXiv:2304.01762* .

Sun, Q., Zhang, W. and Lin, X. (2023), 'Progressive hard negative masking: From global uniformity to local tolerance', *IEEE Transactions on Knowledge and Data Engineering* pp. 1–12.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C. and Isola, P. (2020), What makes for good views for contrastive learning?, *in* H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan

and H. Lin, eds, 'Advances in Neural Information Processing Systems', Vol. 33, Curran Associates, Inc., pp. 6827–6839.

van den Oord, A., Li, Y. and Vinyals, O. (2018), 'Representation learning with contrastive predictive coding'.
URL: *http://arxiv.org/abs/1807.03748*

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y. and Hjelm, R. D. (2019), Deep graph infomax, *in* 'International Conference on Learning Representations'.
URL: *https://openreview.net/forum?id=rklz9iAcKQ*

Wang, T. and Isola, P. (2020), Understanding contrastive representation learning through alignment and uniformity on the hypersphere, *in* H. D. III and A. Singh, eds, 'Proceedings of the 37th International Conference on Machine Learning', Vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 9929–9939.
URL: *https://proceedings.mlr.press/v119/wang20k.html*

Wang, Y. and Yang, Y. (2022), 'Bayesian robust graph contrastive learning'.

Wu, M. and Goodman, N. (2020), 'A simple framework for uncertainty in contrastive learning', *arXiv preprint arXiv:2010.02038* .

Xie, Y., Xu, Z., Zhang, J., Wang, Z. and Ji, S. (2023), 'Self-supervised learning of graph neural networks: A unified review', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **45**(2), 2412–2429.

Yang, M., Isufi, E. and Leus, G. (2022), Simplicial convolutional neural networks, *in* 'ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', pp. 8847–8851.

Yang, M., Isufi, E., Schaub, M. T. and Leus, G. (2021), Finite impulse response filters for simplicial complexes, *in* 'European Signal Processing Conference (EUSIPCO)', Vol. 29.

You, Y., Chen, T., Shen, Y. and Wang, Z. (2021), Graph contrastive learning automated, *in* M. Meila and T. Zhang, eds, 'Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event', Vol. 139 of *Proceedings of Machine Learning Research*, PMLR, pp. 12121–12132.
URL: *http://proceedings.mlr.press/v139/you21a.html*

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z. and Shen, Y. (2020), Graph contrastive learning with augmentations, *in* H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin,

eds, 'Advances in Neural Information Processing Systems', Vol. 33, Curran Associates, Inc., pp. 5812–5823.

Zbontar, J., Jing, L., Misra, I., LeCun, Y. and Deny, S. (2021), Barlow twins: Self-supervised learning via redundancy reduction, *in* M. Meila and T. Zhang, eds, 'Proceedings of the 38th International Conference on Machine Learning', Vol. 139 of *Proceedings of Machine Learning Research*, PMLR, pp. 12310–12320.
URL: *https://proceedings.mlr.press/v139/zbontar21a.html*

Zhang, O., Wu, M., Bayrooti, J. and Goodman, N. (2021), 'Temperature as uncertainty in contrastive learning'.

Zhang, S., Tong, H., Xu, J. and Maciejewski, R. (2019), 'Graph convolutional networks: a comprehensive review', *Computational Social Networks* **6**.

Zhu, Y., Xu, Y., Liu, Q. and Wu, S. (2021), An empirical study of graph contrastive learning, *in* 'Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)'.
URL: *https://openreview.net/forum?id=UuUbIYnHKO*

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S. and Wang, L. (2020), Deep Graph Contrastive Representation Learning, *in* 'ICML Workshop on Graph Representation Learning and Beyond'.
URL: *http://arxiv.org/abs/2006.04131*

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S. and Wang, L. (2021), Graph contrastive learning with adaptive augmentation, *in* 'Proceedings of the Web Conference 2021', WWW '21, Association for Computing Machinery, New York, NY, USA, p. 2069–2080.
URL: *https://doi.org/10.1145/3442381.3449802*

Appendix

A Derivation Graph Filter in the Frequency Domain

Plugging the eigendecomposition of the shift operator $\mathbf{S} = \mathbf{U}\,\mathrm{diag}(\boldsymbol{\lambda})\mathbf{U}^{-1}$ into the expression for the graph convolutional filter, eq. (12), gives:

$$\mathbf{z} = \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x} \tag{47}$$

$$= \sum_{k=0}^{K} h_k \mathbf{U}\,\mathrm{diag}\left(\boldsymbol{\lambda}^{\odot k}\right)\mathbf{U}^{-1}\mathbf{x} \tag{48}$$

$$= \sum_{k=0}^{K} h_k \mathbf{U}\,\mathrm{diag}\left(\boldsymbol{\lambda}^{\odot k}\right)\tilde{\mathbf{x}} \tag{49}$$

where $\boldsymbol{\lambda}^{\odot k} \in \mathbb{C}^N : \left[\boldsymbol{\lambda}^{\odot k}\right]_i := \lambda_i^k$. Then multiplying both sides by $\mathbf{U}^{-1}$ yields:

$$\tilde{\mathbf{z}} = \mathbf{U}^{-1}\mathbf{z} = \sum_{k=0}^{K} h_k\,\mathrm{diag}\left(\boldsymbol{\lambda}^{\odot k}\right)\tilde{\mathbf{x}} \tag{50}$$

B Motivation for the InfoNCE

To see how the InfoNCE originated let us suppose that we have obtained multiple data points $\mathbf{x}_1, ..., \mathbf{x}_M$ with one real (correct) $\mathbf{x}_i \sim p(\mathbf{x} \mid \mathbf{z}')$ and the other ones coming from the marginal (noise) distribution $p(\mathbf{x})$. Here, $\mathbf{z}'$ is the latent variable that constitutes the contrast/context/semantic class that we aim to identify and can be obtained by encoding the context data point $\mathbf{x}'$. If we now want to differentiate the real data point from the noise and let d be the indicator variable for sample $\mathbf{x}_i$ being the correct example, then the related cross-entropy loss would be:

$$\begin{aligned} p\left(d = i \mid \mathbf{x}_1, ..., \mathbf{x}_M, \mathbf{z}'\right) &= \frac{p\left(\mathbf{x}_i \mid \mathbf{z}'\right) \prod_{l \neq i} p\left(\mathbf{x}_l\right)}{\sum_{j=1}^{M} p\left(\mathbf{x}_j \mid \mathbf{z}'\right) \prod_{l \neq j} p\left(\mathbf{x}_l\right)} \\ &= \frac{\frac{p(\mathbf{x}_i|\mathbf{z}')}{p(\mathbf{x}_i)}}{\sum_{j=1}^{M} \frac{p(\mathbf{x}_j|\mathbf{z}')}{p(\mathbf{x}_j)}} \end{aligned} \tag{51}$$

where the density ratio $\frac{p(\mathbf{x}|\mathbf{z}')}{p(\mathbf{x})}$ quantifies how much more likely a sample is given the contrast than under the marginal. This ratio is non-negative and can be (proportionally) approximated via a score function. In the InfoNCE paper (van den Oord et al., 2018), this score function is taken to be:

$$f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right) = \exp\left(\mathbf{z}^T \boldsymbol{\rho}\, \mathbf{z}'\right) \tag{52}$$

where $\mathbf{z} = Q(\mathbf{x})$ and $\boldsymbol{\rho}$ is a matrix that contains learnable weights. This function tends to be large when the embedding of a data point and the contrast point are close in the embedding space. Optimizing the cross-entropy loss based on $f_{\boldsymbol{\rho}}\left(\mathbf{x}, \mathbf{z}'\right)$ thus encourages the model to map data points close to each other that are likely to appear together. Plugging the similarity function in the above equation for the cross-entropy we get:

$$p\left(d = i \mid \mathbf{x}_1, ..., \mathbf{x}_M, \mathbf{z}'\right) = \frac{\frac{p(\mathbf{x}_i|\mathbf{z}')}{p(\mathbf{x}_i)}}{\sum_{j=1}^{M} \frac{p(\mathbf{x}_j|\mathbf{z}')}{p(\mathbf{x}_j)}} \approx \frac{f_{\boldsymbol{\rho}}\left(\mathbf{z}_i, \mathbf{z}'\right)}{\sum_{j=1}^{M} f_{\boldsymbol{\rho}}\left(\mathbf{z}_j, \mathbf{z}'\right)} = \frac{\exp\left(\mathbf{z}_i^T \boldsymbol{\rho}\, \mathbf{z}'\right)}{\sum_{j=1}^{M} \exp\left(\mathbf{z}_j^T \boldsymbol{\rho}\, \mathbf{z}'\right)} \tag{53}$$

If we now take the log-probability and the expectation over the data of the above expression, then the objective we want to minimize is called the InfoNCE:

$$\mathcal{L} = -\mathop{\mathbb{E}}_{X}\left[\log \frac{f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right)}{f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right) + \sum_{\mathbf{x}_j \in X} f_{\boldsymbol{\rho}}\left(\mathbf{z}_j, \mathbf{z}'\right)}\right] \tag{54}$$

whith $M - 1$ negative examples coming from the marginal.

C The InfoNCE As A Bound On The Mutual Information

Appendix B is a prerequisite for the following derivation. Consider the mutual information between the data point $\mathbf{x}$ and the contrast data $\mathbf{x}'$:

$$I(\mathbf{x};\mathbf{x}') \geq I(\mathbf{x};\mathbf{z}') = \sum_{\mathbf{x},\mathbf{z}'} p(\mathbf{x},\mathbf{z}') \log \frac{p(\mathbf{x} \mid \mathbf{z}')}{p(\mathbf{x})} \tag{55}$$

where the bound stems from the basic properties of mutual information (i.e. $I(\mathbf{x};\mathbf{x}') \geq I(\mathbf{x};g(\mathbf{x}'))$ for g being a deterministic function) and we consider the contrast as given. When looking at this expression, we can already realize that it depends on the density ratio that is a vital part of the InfoNCE objective. In fact, when we train the model with this objective we maximise a lower bound on this mutual information. To see this, let us plug the density ration back into the InfoNCE loss:

$$\begin{aligned}
\mathcal{L}_{\mathrm{M}}^{\mathrm{opt}} &= -\mathop{\mathbb{E}}_{X} \log \left[\frac{\frac{p(\mathbf{x}|\mathbf{z}')}{p(\mathbf{x})}}{\frac{p(\mathbf{x}|\mathbf{z}')}{p(\mathbf{x})} + \sum_{\mathbf{x}_j \in X} \frac{p(\mathbf{x}_j|\mathbf{z}')}{p(\mathbf{x}_j)}} \right] \\
&= \mathop{\mathbb{E}}_{X} \log \left[1 + \frac{p\left(\mathbf{x}\right)}{p\left(\mathbf{x} \mid \mathbf{z}'\right)} \sum_{\mathbf{x}_j \in X} \frac{p\left(\mathbf{x}_j \mid \mathbf{z}'\right)}{p\left(\mathbf{x}_j\right)} \right] \\
&= \mathop{\mathbb{E}}_{X} \log \left[1 + \frac{p\left(\mathbf{x}\right)}{p\left(\mathbf{x} \mid \mathbf{z}'\right)} \frac{(M-1)}{(M-1)} \sum_{\mathbf{x}_j \in X} \frac{p\left(\mathbf{x}_j \mid \mathbf{z}'\right)}{p\left(\mathbf{x}_j\right)} \right] \\
&\approx \mathop{\mathbb{E}}_{X} \log \left[1 + \frac{p\left(\mathbf{x}\right)}{p\left(\mathbf{x} \mid \mathbf{z}'\right)} (M-1) \mathop{\mathbb{E}}_{\mathbf{x}_j} \left[\frac{p\left(\mathbf{x}_j \mid \mathbf{z}'\right)}{p\left(\mathbf{x}_j\right)} \right] \right] \\
&= \mathop{\mathbb{E}}_{X} \log \left[1 + \frac{p\left(\mathbf{x}\right)}{p\left(\mathbf{x} \mid \mathbf{z}'\right)} (M-1) \right] \\
&\geq \mathop{\mathbb{E}}_{X} \log \left[\frac{p\left(\mathbf{x}\right)}{p\left(\mathbf{x} \mid \mathbf{z}'\right)} M \right] \ \text{(because } p\left(\mathbf{x} \mid \mathbf{z}'\right) > p\left(\mathbf{x}\right)\text{)} \\
&= -I\left(\mathbf{x},\mathbf{z}'\right) + \log(M).
\end{aligned} \tag{56}$$

where then $I\left(\mathbf{x},\mathbf{z}'\right) \geq \log(M) - \mathcal{L}_{\mathrm{M}}^{\mathrm{opt}}$ is a bound we maximize by minimizing the contrastive loss $\mathcal{L}_{\mathrm{M}}^{\mathrm{opt}}$. The version of the (approximate) proof of this presented here is a small alteration of the one presented by van den Oord et al. (2018). A more general discussion on variational bounds on the mutual information and a proof that does not rely on an approximation can be found in Poole et al. (2019). As a consequence of this bound, we can use a

model trained with the InfoNCE to generate an embedding for a data point $\mathbf{x}$ that (approximately) contains the information we desire as long as we have a way to obtain a datapoint $\mathbf{x}'$ that shares this information with $\mathbf{x}$.

D The ELBO For Contrastive Learning

In the probabilistic model (Aitchison and Ganev, 2023) of contrastive learning we observe two correlated data points $\mathbf{x}$ and $\mathbf{x}'$ generated by two latent variables $\mathbf{z}$, $\mathbf{z}'$. The mapping between the data and the latents is approximated with two encoders $\mathrm{Q}(\mathbf{z} \mid \mathbf{x}, \phi_1)$, $\mathrm{Q}(\mathbf{z} \mid \mathbf{x}, \phi_2)$ where the weights are usually shared, i.e, $\phi = \phi_1 = \phi_2$. To be Bayesian, we learn a distribution over weights of the encoders and over parameters of the similarity function $\boldsymbol{\rho}$. The data points are taken to be independent given the latents which leads to the following decomposition of the joint probability of the statistical model:

$$\begin{aligned} & p\left(\mathbf{x}, \mathbf{x}', \mathbf{z}, \mathbf{z}', \phi_1, \phi_2, \boldsymbol{\rho}\right) \\ & = p\left(\mathbf{x}, \mathbf{x}', \mathbf{z}, \mathbf{z}' \mid \phi_1, \phi_2, \boldsymbol{\rho}\right) p\left(\phi_1, \phi_2, \boldsymbol{\rho}\right) \\ & = p\left(\mathbf{x} | \mathbf{z}, \phi_1\right) p\left(\mathbf{x}' | \mathbf{z}', \phi_2\right) p\left(\mathbf{z}, \mathbf{z}' |, \phi_1, \phi_2, \boldsymbol{\rho}\right) p\left(\phi_1, \phi_2, \boldsymbol{\rho}\right) \end{aligned} \tag{57}$$

Here, $p\left(\boldsymbol{\phi_1}, \boldsymbol{\phi_2}, \boldsymbol{\rho}\right)$ is a prior distribution over the model parameters, $p\left(\mathbf{z}, \mathbf{z}' |, \phi_1, \phi_2, \boldsymbol{\rho}\right)$ a prior over the embeddings and $p\left(\mathbf{x} | \mathbf{z}, \phi_1\right)$, $p\left(\mathbf{x}' | \mathbf{z}', \phi_2\right)$ are likelihoods. The latter can be approximated by using the encoders:

$$\begin{aligned} p\left(\mathbf{x} \mid \mathbf{z}, \phi_1\right) &= \frac{\mathrm{Q}(\mathbf{z} \mid \mathbf{x}, \phi_1)\, p_{\text{true}}\ (\mathbf{x})}{\mathrm{Q}(\mathbf{z} | \phi_1)} \\ p\left(\mathbf{x}' \mid \mathbf{z}', \phi_2\right) &= \frac{\mathrm{Q}(\mathbf{z}' \mid \mathbf{x}', \phi_2)\, p_{\text{true}}\ (\mathbf{x}')}{\mathrm{Q}(\mathbf{z}' | \phi_2)} \end{aligned} \tag{58}$$

where

$$\begin{aligned} \mathrm{Q}\left(\mathbf{z} \mid \boldsymbol{\phi_1}\right) &= \int \mathrm{Q}\left(\mathbf{z} \mid \mathbf{x}, \phi_1\right) p_{\text{true}}\ (\mathbf{x})\, d\mathbf{x} \\ \mathrm{Q}\left(\mathbf{z}' \mid \boldsymbol{\phi_2}\right) &= \int \mathrm{Q}\left(\mathbf{z}' \mid \mathbf{x}', \boldsymbol{\phi_2}\right) p_{\text{true}}\ (\mathbf{x}')\, d\mathbf{x}' \end{aligned} \tag{59}$$

are normalizing constants. The true distribution over the data $p_{\text{true}}\ (\mathbf{x})$ is usually unknown, but we will be able to optimize the model parameters anyways. We continue the derivation by introducing a variational distribution $\mathrm{q}(\boldsymbol{\phi_1}, \boldsymbol{\phi_2}, \boldsymbol{\rho} \mid \boldsymbol{\theta})$ over the parameters which

yields:

$$
\begin{aligned}
&\log p\left(\mathbf{x}, \mathbf{x}'\right) \\
&= \log \int p\left(\mathbf{x}, \mathbf{x}', \mathbf{z}, \mathbf{z}' \mid \boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho}\right) p\left(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho}\right) d\boldsymbol{\phi}_1 \, d\boldsymbol{\phi}_2 \, d\boldsymbol{\rho} \, d\mathbf{z} \, d\mathbf{z}' \\
&= \log \int \mathrm{Q}(\mathbf{z}, \mathbf{z}' \mid \mathbf{x}, \mathbf{x}', \boldsymbol{\phi}_1, \boldsymbol{\phi}_2) \, \mathrm{q}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho} \mid \boldsymbol{\theta}) \\
&\qquad \frac{p\left(\mathbf{x}, \mathbf{x}', \mathbf{z}, \mathbf{z}' \mid \boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho}\right) p\left(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho}\right)}{\mathrm{Q}(\mathbf{z}, \mathbf{z}' \mid \mathbf{x}, \mathbf{x}', \boldsymbol{\phi}_1 \, \boldsymbol{\phi}_2) \, \mathrm{q}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho} \mid \boldsymbol{\theta})} \, d\boldsymbol{\phi}_1 \, d\boldsymbol{\phi}_2 \, d\boldsymbol{\rho} \, d\mathbf{z} \, d\mathbf{z}' \\
&= \log \mathrm{E}_{\mathrm{Q}(\mathbf{z}, \mathbf{z}' \mid \mathbf{x}, \mathbf{x}', \boldsymbol{\phi}_1, \boldsymbol{\phi}_2) \mathrm{q}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho} \mid \boldsymbol{\theta})} \left[\frac{p\left(\mathbf{x}, \mathbf{x}', \mathbf{z}, \mathbf{z}' \mid \boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho}\right) p\left(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho}\right)}{\mathrm{Q}(\mathbf{z}, \mathbf{z}' \mid \mathbf{x}, \mathbf{x}', \boldsymbol{\phi}_1 \, \boldsymbol{\phi}_2) \, \mathrm{q}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho} \mid \boldsymbol{\theta})} \right]
\end{aligned}
\tag{60}
$$

plugging in the approximate likelihoods from eq. (58) into eq. (57) and the result into eq. (60) then gives:

$$
\begin{aligned}
&\log p\left(\mathbf{x}, \mathbf{x}'\right) \\
&= \log \mathrm{E}_{\mathrm{Q}(\mathbf{z}, \mathbf{z}' \mid \mathbf{x}, \mathbf{x}', \boldsymbol{\phi}_1, \boldsymbol{\phi}_2) \mathrm{q}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho} \mid \theta)} \left[\frac{p\left(\mathbf{z}, \mathbf{z}' \mid \boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho}\right) p\left(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho}\right)}{\mathrm{Q}\left(\mathbf{z}' \mid \boldsymbol{\phi}_2\right) \mathrm{Q}\left(\mathbf{z} \mid \boldsymbol{\phi}_1\right) \; \mathrm{q}\left(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho} \mid \boldsymbol{\phi}_2\right)} \right] + \text{const} \\
&\geq \mathrm{E}_{\mathrm{Q}(\mathbf{z}, \mathbf{z}' \mid \mathbf{x}, \mathbf{x}', \boldsymbol{\phi}_1, \boldsymbol{\phi}_2) \mathrm{q}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho} \mid \theta)} \left[\log \frac{p\left(\mathbf{z}, \mathbf{z}' \mid \boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho}\right) p\left(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho}\right)}{\mathrm{Q}\left(\mathbf{z}' \mid \boldsymbol{\phi}_2\right) \mathrm{Q}\left(\mathbf{z} \mid \boldsymbol{\phi}_1\right) \; \mathrm{q}\left(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho} \mid \boldsymbol{\theta}\right)} \right]
\end{aligned}
\tag{61}
$$

where we used Jensen's inequality and the constant term comes from the true data distributions. Modelling the weights $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho}$ as independent from each other this can be rewritten as:

$$
\begin{aligned}
\log p\left(\mathbf{x}, \mathbf{x}'\right) \geq \; & \mathrm{E}_{\mathrm{Q}(\mathbf{z}, \mathbf{z}' \mid \mathbf{x}, \mathbf{x}', \boldsymbol{\phi}_1, \boldsymbol{\phi}_2) \mathrm{q}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho} \mid \boldsymbol{\theta})} \left[\log \frac{p\left(\mathbf{z}, \mathbf{z}' \mid \boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\rho}\right)}{\mathrm{Q}\left(\mathbf{z}' \mid \boldsymbol{\phi}_2\right) \mathrm{Q}\left(\mathbf{z} \mid \boldsymbol{\phi}_1\right)} \right] \\
& - KL(\mathrm{q}(\boldsymbol{\phi}_1 \mid \boldsymbol{\theta}) || P(\boldsymbol{\phi}_1)) \\
& - KL(\mathrm{q}(\boldsymbol{\phi}_2 \mid \boldsymbol{\theta}) || P(\boldsymbol{\phi}_2)) \\
& - KL(\mathrm{q}(\boldsymbol{\rho} \mid \boldsymbol{\theta}) || P(\boldsymbol{\rho}))
\end{aligned}
\tag{62}
$$

This is an ELBO for a Bayesian Contrastive Learning model. In the next section we will see how the right choice of prior over the embeddings leads to the InfoNCE (in a deterministic setting).

E The (Deterministic) ELBO Is The InfoNCE For A Specific Prior

For a deterministic encoder (i.e. no weight uncertainty is taken into account) and averaging over the data the ELBO for contrastive learning is (Aitchison and Ganev, 2023):

$$\log p\left(\mathbf{x}, \mathbf{x}'\right) \geq \mathrm{E}_{\mathrm{Q}_{\boldsymbol{\phi_1}, \boldsymbol{\phi_2}}(\mathbf{z}, \mathbf{z}')}\left[\log \frac{p\left(\mathbf{z}, \mathbf{z}'\right)}{\mathrm{Q}_{\boldsymbol{\phi_1}}(\mathbf{z})\, \mathrm{Q}_{\boldsymbol{\phi_2}}(\mathbf{z}')}\right] + \text{const.} \tag{63}$$

This is equivalent to first part in eq. (62) when no distribution over the parameters is taken into account. Based on this, we choose the same weights for both encoders ($\boldsymbol{\phi} = \boldsymbol{\phi}_1 = \boldsymbol{\phi}_2$) and plug in the following choice for the prior

$$\begin{aligned} p(\mathbf{z}) &= \mathrm{Q}_{\boldsymbol{\phi}}(\mathbf{z}) \\ p\left(\mathbf{z}' \mid \mathbf{z}\right) &= \frac{1}{Z_{\boldsymbol{\phi}, \boldsymbol{\rho}}(\mathbf{z})} \mathrm{Q}_{\boldsymbol{\phi}}\left(\mathbf{z}'\right) f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right) \end{aligned} \tag{64}$$

with the normalising constant

$$Z_{\boldsymbol{\rho}, \boldsymbol{\phi}}(\mathbf{z}) = \int \mathrm{Q}_{\boldsymbol{\phi}}\left(\mathbf{z}'\right) f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right) d\mathbf{z}' \tag{65}$$

. Notably, this is a prior that is parametrized by the encoding neural network, a frequently used trick in the literature (Aitchison and Ganev, 2023). The construction then gives us the ELBO for the InfoNCE:

$$\mathcal{L}_{\text{InfoNCE}}\left(\boldsymbol{\rho}, \boldsymbol{\phi}\right) = \mathrm{E}_{\mathrm{Q}_{\boldsymbol{\phi}}(\mathbf{z}, \mathbf{z}')}\left[\log \frac{\mathrm{Q}_{\boldsymbol{\phi}}(\mathbf{z}) \frac{1}{Z_{\boldsymbol{\rho}, \boldsymbol{\phi}}(\mathbf{z})} \mathrm{Q}_{\boldsymbol{\phi}}\left(\mathbf{z}'\right) f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right)}{\mathrm{Q}_{\boldsymbol{\phi}}(\mathbf{z}) \mathrm{Q}_{\boldsymbol{\phi}}\left(\mathbf{z}'\right)}\right] + \text{const} \tag{66}$$

by cancelling we get:

$$\mathcal{L}_{\text{InfoNCE}}\left(\boldsymbol{\rho}, \boldsymbol{\phi}\right) = \mathrm{E}_{\mathrm{Q}_{\boldsymbol{\phi}}(\mathbf{z}, \mathbf{z}')}\left[\log \frac{f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right)}{Z_{\boldsymbol{\rho}, \boldsymbol{\phi}}(\mathbf{z})}\right] + \text{const} \tag{67}$$

$$= \mathrm{E}_{\mathrm{Q}_{\boldsymbol{\phi}}(\mathbf{z}, \mathbf{z}')}\left[\log \frac{f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right)}{\int \mathrm{Q}_{\boldsymbol{\phi}}\left(\mathbf{z}'\right) f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right) d\mathbf{z}'}\right] + \text{const} \tag{68}$$

at this point we already note the similarity with eq. (19), which is the (negative) finite-sample estimator. Formally this can see be seen by rewriting the above expression as:

$$\mathcal{L}_{\text{InfoNCE}}\left(\boldsymbol{\rho}, \boldsymbol{\phi}\right) = \mathrm{E}_{\mathrm{Q}_{\boldsymbol{\phi}}(\mathbf{z}, \mathbf{z}')}\left[\log f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right)\right] - \mathrm{E}_{\mathrm{Q}_{\boldsymbol{\phi}}(\mathbf{z})}\left[\log \mathrm{E}_{\mathrm{Q}_{\boldsymbol{\phi}}(\mathbf{z}')}\left[f_{\boldsymbol{\rho}}\left(\mathbf{z}, \mathbf{z}'\right)\right]\right] + \text{ const.} \tag{69}$$

This is up to a constant equivalent to the infinite-sample InfoNCE objective as derived by (Wang and Isola, 2020, Li et al., 2021).

F Derivation FIR Frequency Response

Note that that we can decompose a filter $\mathbf{H}$ as:

$$\mathbf{H} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\top \tag{70}$$

and then compute the frequency response as:

$$h(\mathbf{\Lambda}) = \mathbf{U}^\top \mathbf{H}\mathbf{U} \tag{71}$$

Now, consider the Finite Impulse Response (FIR) Filter for edge flows (Yang et al., 2021):

$$\mathbf{H} = \epsilon\mathbf{I} + \sum_{l_1=0}^{L_1} \alpha_{l_1} (\mathbf{L}_{1,l})^{l_1} + \sum_{l_2=0}^{L_2} \beta_{l_2} (\mathbf{L}_{1,u})^{l_2} \tag{72}$$

where ϵ, $\alpha = \{\alpha_1, ..., \alpha_{L_1}\}$ and $\beta = \{\beta_1, ..., \beta_{L_2}\}$ are the filter coefficients and L_1, L_2 specify the order of the filter. To derive the frequency response we first rewrite $\mathbf{H}$:

$$\begin{aligned}
\mathbf{H} &= \epsilon\mathbf{I} + \sum_{l_1=0}^{L_1} \alpha_{l_1} (\mathbf{U}_1 \operatorname{blkdiag}(\mathbf{o}, \mathbf{\Lambda}_\mathrm{G}, \mathbf{o})\, \mathbf{U}_1^\top)^{l_1} \\
&\quad + \sum_{l_2=0}^{L_2} \beta_j (\mathbf{U}_1 \operatorname{blkdiag}(\mathbf{o}, \mathbf{o}, \mathbf{\Lambda}_\mathrm{C})\, \mathbf{U}_1^\top)^{l_2} \\
&= \epsilon\mathbf{I} + \sum_{l_1=0}^{L_1} \alpha_{l_1} \mathbf{U}_1 \operatorname{blkdiag}(\mathbf{o}, \mathbf{\Lambda}_\mathrm{G}, \mathbf{o})^{\odot l_1}\, \mathbf{U}_1^\top \\
&\quad + \sum_{l_2=0}^{L_2} \beta_{l_2} \mathbf{U}_1 \operatorname{blkdiag}(\mathbf{o}, \mathbf{o}, \mathbf{\Lambda}_\mathrm{C})^{\odot l_2}\, \mathbf{U}_1^\top
\end{aligned}$$

where, we used that $\mathbf{L}_{1,\ell} = \mathbf{U}_1 \operatorname{blkdiag}(\mathbf{o}, \mathbf{\Lambda}_\mathrm{G}, \mathbf{o})\, \mathbf{U}_1^\top$ and $\mathbf{L}_{1,\mathrm{u}} = \mathbf{U}_1 \operatorname{blkdiag}(\mathbf{o}, \mathbf{o}, \mathbf{\Lambda}_\mathrm{C})\, \mathbf{U}_1^\top$. This then allows us to compute the frequency response:

$$h(\mathbf{\Lambda}) = \mathbf{U}_1^\top \mathbf{H}\mathbf{U}_1 \tag{73}$$

$$= \epsilon\mathbf{I} + \sum_{l_1=0}^{L_1} \alpha_{l_1} \operatorname{blkdiag}(\mathbf{o}, \mathbf{\Lambda}_\mathrm{G}, \mathbf{o})^{\odot l_1} + \sum_{l_2=0}^{L_2} \beta_{l_2} \operatorname{blkdiag}(\mathbf{o}, \mathbf{o}, \mathbf{\Lambda}_\mathrm{C})^{\odot l_2} \tag{74}$$

and isolating the response for an individual eigenvalue we get:

$$h(\lambda_i) = \begin{cases} \epsilon, \text{ for } \lambda_i \in \mathbf{\Lambda}_\mathrm{H} \\ \epsilon + \sum_{l_1=1}^{L_1} \alpha_{l_1} \lambda_i^{l_1}, \text{ for } \lambda_i \in \mathbf{\Lambda}_\mathrm{G} \\ \epsilon + \sum_{l_2=1}^{L_2} \beta_{l_2} \lambda_i^{l_2}, \text{ for } \lambda_i \in \mathbf{\Lambda}_\mathrm{C} \end{cases} \tag{75}$$

G Spectral Augmentations via Simplicial Filters

One way to influence information in the spectral domain in a controlled way is by using a suitable filter and designing its parameters such that it removes irrelevant frequencies. To obtain a stochastic mechanism we consider a distribution over the filter parameters and sample different filters to generate augmentations of the original data point. For Finite Impulse Response Filter, eq. (39), with normally distributed parameters we can write the resulting augmentation of a data point as:

$$\mathbf{x}' = \mathcal{T}(\mathbf{x}) = \mathbf{H}^{FIR}\mathbf{x} = \left(\epsilon\mathbf{I} + \sum_{l_1=0}^{L_1} \alpha_{l_1}(\mathbf{L}_{1,l})^{l_1} + \sum_{l_2=0}^{L_2} \beta_{l_2}(\mathbf{L}_{1,u})^{l_2}\right)\mathbf{x} \tag{76}$$

where we sample $\epsilon \sim \mathcal{N}(\mu_\epsilon, \sigma_\epsilon^2)$, $\alpha_{l_1} \sim \mathcal{N}(\mu_{\alpha_{l_1}}, \sigma^2_{\alpha_{l_1}})$, $\beta_{l_2} \sim \mathcal{N}(\mu_{\beta_{l_2}}, \sigma^2_{\beta_{l_2}})$.

To see how such an augmentation affects the spectral components of the data, recall the deterministic frequency response of a FIR filter given in eq. (40). Based on this, the frequency response for normally distributed parameters is normally distributed as well as the sum of independent normals is again normal. This gives:

$$h(\lambda_i) \sim \begin{cases} \mathcal{N}(\mu_\epsilon, \sigma_\epsilon^2), \text{ for } \lambda_i \in \mathbf{\Lambda}_\text{H} \\ \mathcal{N}(\mu_g, \sigma_g^2), \text{ for } \lambda_i \in \mathbf{\Lambda}_\text{G} \\ \mathcal{N}(\mu_c, \sigma_c^2), \text{ for } \lambda_i \in \mathbf{\Lambda}_\text{C} \end{cases} \tag{77}$$

with $\mu_g = \mu_\epsilon + \sum_{l_1=1}^{L_1} \mu_{\alpha_{l_1}} \lambda_i^{l_1}$, $\sigma_g = \sigma_\epsilon^2 + \sum_{l_1=1}^{L_1} \sigma^2_{\alpha_{l_1}} (\lambda_i^{l_1})^2$ for the gradients and $\mu_c = \mu_\epsilon + \sum_{l_2=1}^{L_2} \mu_{\beta_{l_1}} \lambda_i^{l_2}$, $\sigma_c = \sigma_\epsilon^2 + \sum_{l_2=1}^{L_2} \sigma^2_{\beta_{l_2}} (\lambda_i^{l_2})^2$ for curl eigenvalues. From this we can see that the parameters of the distributions can be chosen/tuned such that in probability we obtain an augmentation that preserves or changes the signal's gradients, curl or harmonic parts. To be more specific, consider $\mathbb{E}\left[(h(\lambda_i) - 1)^2\right]$, the expected squared distance of the frequency response to 1. The distance to 1 is appropriate, because if $h(\lambda_i) = 1$ then the specific frequency of the signal is not changed. This expression can be expanded as:

$$\begin{aligned} \mathbb{E}\left[(h(\lambda_i) - 1)^2\right] &= \mathbb{E}\left[h(\lambda_i)^2\right] - 2\mathbb{E}\left[h(\lambda_i)\right] + 1 \\ &= \mathbb{V}\left[h(\lambda_i)\right] + \mathbb{E}\left[h(\lambda_i)\right]^2 - 2\mathbb{E}\left[h(\lambda_i)\right] + 1 \end{aligned} \tag{78}$$

and we can use this to calculate how we should pick the parameters of the normal distributions from which we sample the filters. For instance, consider a FIR filter with $L_1 = 1$, $L_2 = 1$ to reduce the number of parameters, and $\mu_\epsilon = 1$, $\mu_{\alpha_{l_1}} = 0$, $\mu_{\beta l_1} = 0$ to keep the change in response to be symmetric around 1 (while small in the harmonic component).

Table 3: Test Accuracies for the FIR augmentation (Appendix F).

Model	Trajectory Task	Ocean Drifters
SCL_{FIR}	94.4 ± 0.8	82.0 ± 1.6

This then results in the following expected squared distances for the curl, gradient and harmonic frequencies:

$$\mathbb{E}\left[\left(h\left(\lambda_i\right)-1\right)^2\right] \sim \begin{cases} \sigma_\epsilon^2 \text{ for } \lambda_i \in \mathbf{\Lambda}_{\mathrm{H}} \\ \sigma_\epsilon^2+\sigma_{\alpha_1}^2, \text{ for } \lambda_i \in \mathbf{\Lambda}_{\mathrm{G}} \\ \sigma_\epsilon^2+\sigma_{\beta_1}^2, \text{ for } \lambda_i \in \mathbf{\Lambda}_{\mathrm{C}} \end{cases} \tag{79}$$

By picking suitable standard deviations for the filter parameters we can thus, in expectation, induce the desired change in the different frequency components of the Hodge Decomposition. We have designed an interpretable stochastic augmentation method that can be used to,in probability, generate positive examples with specific spectral properties. We have conducted experiments with augmentation and the results can be found in Table 3. The spectrally optimized dropout augmentation from section 8.2.1 outperforms the Finite Impulse Response (FIR) filter augmentation. A possible explanation for the better performance of the first is that it is able to generate examples that are both topologically and spectrally similar to the anchor, while the FIR approach focuses solely on spectral utility. Nevertheless it should be highlighted that the filter augmentation comes with compelling theoretical properties as its frequency response allows for the derivation of bounds and guarantees that enhance our understanding of its operational scope and limitations.

Table 4: Test Accuracies for the Proteins and MSRC21 Datasets.

Model	Proteins	MSRC-21
GIN (supervised)	76.2 ± 0.1	89.2 ± 0.2
SCNN (supervised)	74.1 ± 0.6	86.4 ± 0.9
SCL_{EFM}	73.1 ± 0.3	84.1 ± 0.9

H Additional Experiments on Lifted Graphs

In addition to the trajectory prediction tasks we also evaluate the standard contrastive learner without spectral optimization from section 8.1 on two graph classification tasks for which a lifting transformation to edges is performed (Bodnar et al., 2021). That is, an edge value is defined as the mean value of two adjacent nodes. We use the a protein dataset (Proteins) and a graph-based computer vision dataset (MSRC-21) from the TU Graph Benchmark (Morris et al., 2020). The Protein dataset contains 1113 examples and the MSRC-21 dataset 591 samples where we use 10% of the data for training, 10% as validation set and 80% for the final evaluation.For the network architecture for the SCNN we follow the settings from Bodnar et al. (2021) (who propose a supervised model) and use a hidden-layer of size 64, Tanh-activations for the trajectory tasks and ELU activations for the lifted graph prediction.

On the lifted graph classification tasks , we are not able to outperform the baseline GIN with a simplicial architecture, neither in the supervised nor in the contrastive learning setting (Table 4). The reason for this is likely the loss of information resulting from the lifting transformation, which involves computing the mean and processing data only on the edges. In future works on graph classification this could be resolved by using an architecture that takes information from edges, nodes and triangles, such as in the work of Bodnar et al. (2021). In that way, the information from the nodes could still be included.

I Additional Results VGCL & CMDS

Figure 16: (Left) Retention Curves on Pubmed for a InfoNCE model. (Right) Retention Curves on Pubmed for VI-InfoNCE.

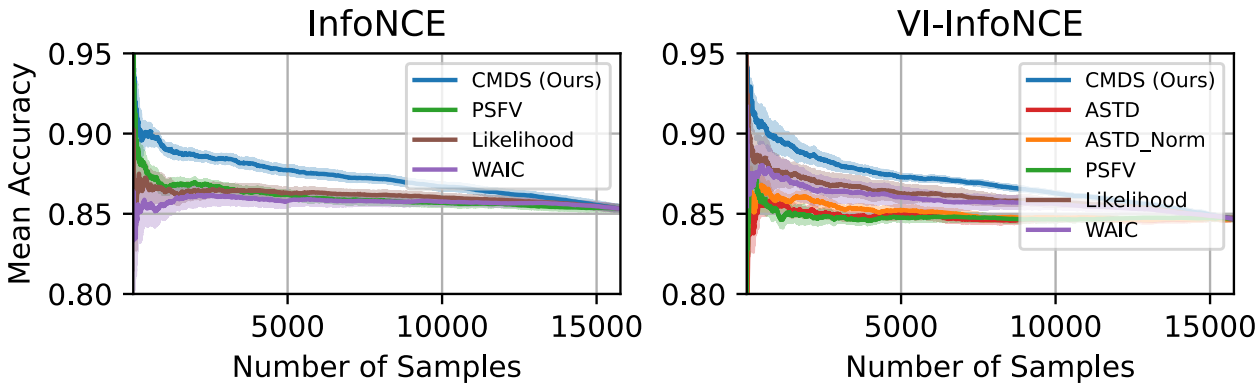

Figure 17: (Left) Retention Curves on Cora for VI-InfoNCE. (Right) Retention Curves on Cora for VGCL.