Self-Supervised Learning of Musical Representations Using VICReg

A comprehensive study of the VICReg loss function for self-supervised representation learning in the music domain

Master's thesis in MPENM* and MPCAS†

CODY HESSE*
SEBASTIAN LÖF†

DEPARTMENT OF ARCHITECTURE AND CIVIL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Self-Supervised Learning of Musical Representations Using VICReg

A comprehensive study of the VICReg loss function for self-supervised representation learning in the music domain

CODY HESSE
SEBASTIAN LÖF

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Architecture and Civil Engineering
Division of Applied Acoustics
Audio Technology Group
Chalmers University of Technology
Gothenburg, Sweden 2023

Self-Supervised Learning of Musical Representations Using VICReg
A comprehensive study of the VICReg loss function for self-supervised representation learning in the music domain
CODY HESSE
SEBASTIAN LÖF

Supervisors: Carlos Lordelo & Carl Thomé , Epidemic Sound
Examiner: Jens Ahrens, Department of Architecture and Civil Engineering

Master's Thesis 2023
Department of Architecture and Civil Engineering
Division of Applied Acoustics
Audio Technology Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: An image depicting self-supervised learning of audio waveforms, generated by a self-supervised learning-based method

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Self-Supervised Learning of Musical Representations Using VICReg
A comprehensive study of the VICReg loss function for self-supervised representation learning in the music domain
CODY HESSE, SEBASTIAN LÖF
Department of Architecture and Civil Engineering
Chalmers University of Technology

Abstract

Self-supervised learning has emerged as a promising method for learning informative representations suitable for many machine learning tasks. However, while self-supervised representation learning has been instrumental in various fields, its significance in music information retrieval has only recently gained momentum. This thesis investigates the potential of the VICReg loss function for self-supervised learning in the music domain by comparing its performance against the established CLMR model. Following the evaluations performed in CLMR, we train our VICReg model on the publically available Free Music Archive and GTZAN datasets. We then evaluate the learned representation on the downstream task of music classification on the MagnaTagATune dataset by training a linear logistic classifier and a two-layer MLP classifier atop the representations generated by a frozen, pre-trained VICReg model. In our transfer learning experiments, VICReg achieves a ROC-AUC score of 89.15 and a PR-AUC score of 35.85 compared to 88.12 and 33.83, respectively, as achieved by CLMR, showing that VICReg demonstrates a competitive performance compared to CLMR. With more robust training and further tuning, we believe that VICReg can achieve superior performance compared to established loss functions for self-supervised representation learning in the music domain and advocate continued exploration in this direction.

Keywords: Self-supervised learning, Contrastive learning, Music Information Retrieval, Representation learning, VICReg, CLMR, SampleCNN.

Acknowledgements

Firstly, we would like to thank our supervisors, Carlos Lordelo and Carl Thomé, for supporting us with excellent knowledge and enthusiasm throughout our stay with Epidemic Sound. Your invaluable feedback has helped us streamline our work and align on essential tasks. This project would not have been the same without you. Moreover, we would like to thank all our fantastic colleagues at Epidemic Sound for the many engaging conversations, ping-pong games, and for keeping our work fun even in times of great stress. We are truly thankful for the experience and grateful for the people we have met.

In addition, we want to thank our examiner, Professor Jens Ahrens, for his trust, support, and guidance throughout the creation of this thesis. Your excitement for this work has motivated us greatly.

We would also like to acknowledge our friends, colleagues, and professors from Chalmers University BSc program in Engineering Physics and our respective master's degrees.

Finally, we would like to thank our families and girlfriends for their love and support. Thank you for always believing in us.

Cody Hesse & Sebastian Löf, Gothenburg, 05 2023

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ANN	Artificial Neural Network
BYOL-A	Bootstrap Your Own Latent for Audio
CLMR	Contrastive Learning of Musical Representations
CNN	Convolutional Neural Network
DML	Deep Metric Learning
FMA	Free Music Archive
JEA	Joint Embedding Architecture
MIR	Music Information Retrieval
MLP	Multilayer Perceptron
MSE	Mean Squared Error
MTAT	MagnaTagATune
NT-Xent	Normalized Temperature-scaled Cross Entropy Loss
PR-AUC	Area Under the Precision-Recall Curve
ReLU	Rectified Linear Unit
ROC-AUC	Area Under the Receiver Operating Characteristic Curve
SSL	Self-Supervised Learning
VICReg	Variance-Invariance-Covariance Regularization

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables used throughout this thesis.

Numerical Objects

x	A scalar
$\mathbf{x}$	A vector
$\mathbf{X}$	A matrix
x_i	The i-th element of the vector $\mathbf{x}$
$\mathbf{X}^{(\ell)}$	Matrix $\mathbf{X}$ at layer ℓ of a neural network
$\mathbf{x}^{(\ell)}$	Vector $\mathbf{x}$ at layer ℓ of a neural network

Functions and Operators

$f(\cdot)$	A funcion f operating on $\cdot$
$\log(\cdot)$	The natural logarithm with base e
$\log_2(\cdot)$	The logarithm with base 2
$\exp(\cdot)$	The exponential function
$(\cdot)^T$	The transpose of a vector or matrix
$\left\| \cdot \right\|_2$	ℓ_2 norm
$f : \mathbf{x} \longmapsto \mathbf{y}$	A function that maps input $\mathbf{x}$ to output $\mathbf{y}$
$\sum_i^n$	Summation over the collection of elements $i = 1, \ldots, n$
$\nabla_{\mathbf{x}}\mathbf{y}$	The gradient of $\mathbf{y}$ with respect to $\mathbf{x}$
$\mathcal{A} \cup \mathcal{B}$	The union of sets $\mathcal{A}$ and $\mathcal{B}$ (containing all elements from set $\mathcal{A}$ and set $\mathcal{B}$)

Parameters

λ	Weight parameter that modulates the influence of the invariance term, $s(\mathbf{Z}, \mathbf{Z}')$ in the VICReg loss
μ	Weight parameter that modulates the influence of the variance term, $v(\mathbf{Z})$ in the VICReg loss
ν	Weight parameter that modulates the influence of the covariance term, $c(\mathbf{Z})$ in the VICReg loss

Sets

$\mathcal{P}$	Set of positive pairs in the NT-Xent loss
$\mathcal{T}$	Set of data augmentations
$\mathcal{D}$	Dataset containing music tracks

Contents

List of Figures

List of Tables

1
Introduction

The last decades have witnessed an explosion of interest and advancement in machine learning, fueled by increasing amounts of data and computational power. Traditional supervised learning paradigms, where a model learns to map input data to known labels, have given rise to multiple applications across various fields. However, they have a significant limitation: the need for large amounts of labeled data. Given the challenges associated with obtaining and curating such datasets, an alternate learning paradigm has garnered much attention in recent years — self-supervised learning (SSL).

Self-supervised learning is a method where models learn to extract informative features or representations from data without any explicit labels [1]. SSL algorithms can capture meaningful patterns by defining pretext tasks based on the input data to produce descriptive and intelligible representations, facilitating subsequent tasks such as classification, detection, or prediction [2]. The strength of SSL lies in its ability to harness the vast amounts of unlabeled data available, sidestepping the labor-intensive manual annotation process.

Many companies and research groups have successfully employed SSL methods within the *Deep Metric Learning* (DML) family in machine learning for audio representation learning [3, 4]. Originating from the idea of *contrastive loss* introduced in 2005 [5], DML models are designed to encourage similarity between transformed versions of the same input. In practice, contrastive learning achieves such similarity by generating an embedding space where *positive pairs*, i.e., transformed pairs originating from the same source, are close and *negative pairs* distant. In a similar approach, *triplet loss* uses a *query* value, a positive sample, and a negative sample to minimize the distance between the query and the positive sample and maximize the distance to the negative sample [6]. Furthermore, adaptations of SSL methods from other modalities, such as the Bootstrap Your Own Latent for Audio (BYOL-A) approach adapted from computer vision, have proven effective in the audio domain [7, 4, 8], underscoring the potentially cross-modal applicability of SSL techniques.

However, while SSL methods have brought significant advances, there remains the challenge of information collapse, a phenomenon where the learned representations tend to collapse into a single point or a smaller subspace, reducing their discriminative power. In order to address information collapse, complex solutions, such as batch or feature-wise normalization, output quantization, usage of stop gradients, or memory banks, are often required [2].

The recently introduced VICReg (Variance-Invariance-Covariance Regularization) loss function [9] presents the potential for addressing information collapse without considering complex model architectures or increased computational requirements. This loss function encourages the model to learn invariance to transformations in input data while avoiding collapse by maintaining a minimum variance and minimizing the off-diagonal elements in the covariance of each batch. While initially evaluated in the image domain, its principles hold promising implications for audio representation learning, prompting the exploration undertaken in this thesis.

In the evolving landscape of machine learning, the potential of self-supervised learning, particularly VICReg, is yet to be fully realized in music and audio understanding. This study compares VICReg with another well-established model in this domain, the Contrastive Learning of Musical Representations (CLMR) [4]. While CLMR, a SimCLR-based method [10], has already demonstrated efficacy for self-supervised audio representation tasks, the properties of VICReg introduce an intriguing point of exploration in the pursuit of robust audio representations.

In this thesis, we begin with an extensive literature review, analyzing and contextualizing the current state of SSL in audio and the role of VICReg and CLMR in this domain. This review provides a broad picture of how SSL has evolved in Music Information Retrieval (MIR). Following this, we delve into the specifics of our experiments, designed to explore various aspects of the VICReg and CLMR models. Our experiments include comprehensive analyses of model convergence, the influence of representation size, and the effectiveness of transfer learning. We conduct these experiments across the same conditions for both models to ensure a fair comparison and follow the experimental setup proposed in CLMR [4].

This thesis is not merely contributing to the literature on SSL in audio representation learning. It is also an exploration of the potential modal adaptation of VICReg from the image domain to the music domain. We hope that the insights from this study can pave the way for more sophisticated audio representation learning methods and enhance the understanding of music and audio data through machine learning.

1.1 Problem formulation

In recent years, multiple studies have shown that contrastive self-supervised learning is an effective way to learn musical representations suitable for downstream tasks such as music tagging, genre classification, and chord recognition [4, 11, 3]. This is often achieved by leveraging a contrastive loss function to minimize the distance between matching data and maximize the distance between non-matching data in an embedding space. Most currently available contrastive losses, however, innately suffer from high memory costs and potentially lead to a dimensional collapse in the generated representations [12, 2, 9]. Hence, following recent developments in the image domain, this study investigates the VICReg loss in the MIR domain, as VICReg has been shown to mitigate the risk of collapse while improving memory

efficiency [9]. Specifically, we aim to reproduce the study which introduced CLMR and, consequently, the normalized temperature scaled cross-entropy loss (*NT-Xent*) loss [4] to the music domain to effectively evaluate and compare VICReg losses to an established model in the music domain.

1.2 Purpose and Aim

The purpose of this thesis is to implement VICReg for the music domain and explore whether it could be used as an effective alternative to NT-Xent loss in self-supervised learning. We perform a comparative analysis between VICReg and CLMR on multiple evaluation tasks such as convergence, representation size, and transfer learning. We also conduct hyperparameter exploration for the VICReg loss.

To fulfill the intended scope, the aim of this thesis is to provide answers to the following questions:

1. Given the same model backbone and training data, how does VICReg compare to CLMR?
2. For different representation dimensions, how does VICReg compare to CLMR in downstream tagging tasks?

1.3 Scope and delimitations

Since the upsurge of attention models in 2017, many audio and MIR benchmarks have been topped by models with ViT or other transformer-based backbones [13, 14]. Despite this recent increase in state-of-the-art transformer results, we have built our model entirely based on the SampleCNN [15] backbone used in CLMR [4]. This is partly to allow for a fair comparison between the results achieved by VICReg and early results using NT-Xent, and partly due to the increased computational cost imposed by other common models, such as large transformer models, which would have forced less extensive evaluation of VICReg performance in various settings. Many backbone architectures could, however, benefit from VICReg loss, which is left for future work.

While we aim to explore some essential hyperparameter tuning, some parameters were kept according to the settings in their respective original publications. In particular, we perform our experiments using the same weight distribution for each of the Variance, Invariance, and Covariance loss components of VICReg as the original publication [9].

The scope of this study is deliberately confined to training our models exclusively on the Free Music Archive (FMA) medium dataset [16] and GTZAN [17] datasets, with evaluations performed solely on the MagnaTagATune [18] dataset. While many other data repositories exist, such as the Million Song Dataset (MSD) [19], NSynth [20], and the McGill Billboard dataset [21], we have selected these particular datasets due

to their public accessibility and to facilitate fair comparison with the experiments carried out in the CLMR study. Exploring the VICReg model's performance across broader datasets, such as HARES [22], which integrates environmental, speech, and music audio data, could offer valuable insights into the model's versatility in representation generalization. However, we leave this as an opportunity for future research.

Although there are arguably infinite ways to determine if two musical tracks are similar, we limit our scope to only study musical similarity as it is commonly defined in the self-supervised MIR setting. This means that two audio tracks are to be deemed similar if they are close in proximity within the embedding space generated by the model backbone. In practice, imposing similarity, in our case, is equivalent to placing data in such a way that closeness improves linear separability for downstream tasks. Consequently, we limit this study to measuring the separability of our data using only a single or, at most, two-layer MLP for downstream task evaluation. Intuitively, this means groups should be distant enough in the embedding space to be linearly separable. While performance could benefit from introducing more complex model heads, allowing the model to find intricate patterns within clusters, we focus on smaller models to favor embedding spaces which contain intuitively and effectively separated clusters.

Finally, we limit this study to only evaluating our models' performance on downstream tasks using the Area under the Receiver-Operator Curve (ROC-AUC) and Precision-Recall Area under Curve (PR-AUC). While many other popular methods exist to evaluate audio classification problems, such as Mean Average Precision (mAP), we chose to limit our model to ROC-AUC and PR-AUC to comply with the evaluation performed in CLMR.

1.4 Related Work

Self-supervised learning has become a leading approach in various domains, with audio representation learning being no exception. A considerable number of studies have proposed different techniques, pushing state-of-the-art performances on various tasks.

1.4.1 Self-Supervised Representation Learning in Audio

Notably, the Wav2Vec 2.0 model [23], and its various derivatives have shown impressive results in self-supervised audio representation tasks. Transformer-based models have gained prominence for speech representations [24], while conformer-based models have been utilized for general audio representations [25], achieving state-of-the-art results on the AudioSet dataset [26] for audio-only self-supervised learning [27]. Moreover, the Contrastive Audio-Visual Masked Auto-Encoder (CAV-MAE) achieved state-of-the-art results on AudioSet by leveraging contrastive learning on audio-and-video modalities.

In a similar vein, KGenre has successfully utilized metadata in conjunction with raw audio, incorporating a GNN-based knowledge graph to guide representation learning. This approach resulted in state-of-the-art results on genre classification on the Free Music Archive dataset [11].

Other studies have focused on adapting loss functions that have proven successful in other modalities, such as images, to the audio domain. An example is the BYOL for Audio (BYOL-A) [7] approach, which introduced an audio adaptation of the BYOL [28] loss function. Likewise, Wang et al. proposed the normalizer-free Slowfast NFNet [29], a 63M parameter model trained on the SimCLR-loss used in CLMR. This model, constructed on the foundations of several architectural improvements in the audio domain, achieved state-of-the-art results across multiple audio tasks.

The SELFIE method also leveraged a BYOL encoder to achieve state-of-the-art results when averaged over various MIR tasks. This approach introduced a diversity loss and utilized the VICReg decorrelation term for information maximization, demonstrating the potential of the VICReg approach in audio tasks[30].

Furthermore, the success of leveraging triplet loss for audio representation learning is worth mentioning. For instance, Thomé et al. built an audio-similarity search engine using triplet loss with a ConvNet encoder [3], while the Multi-modal Fusion Network for Emotion Recognition in Conversation (M2FNet) combined triplet loss with the variance loss used in VICReg to achieve state-of-the-art results in Emotion Recognition in Conversations (ERC) tasks [29].

1.4.2 Contrastive Variations

In contrastive variations, studies such as Contrastive Learning of Auditory Representations (CLAR) [8] and COntrastive Learning for Audio (COLA) [31] have shown promise. Both employed a SimCLR-based NT-Xent loss, similar to CLMR. CLAR utilized 1D and 2D ResNet-18 backbones to process both raw audio waveforms and Mel spectrograms as input, whereas COLA relied entirely on Mel Spectrograms as input and used bilinear comparisons as a similarity metric.

1.4.3 Modal adaptations of VICReg

The application of VICReg in modalities beyond images has been examined in other studies. Karnan et al. used a variation of VICReg loss on a combination of image and sensor input for self-supervised terrain representation from robot experience, achieving state-of-the-art results [32]. Similarly, Xu et al. used the Variance and Covariance regularizers from the VICReg loss to penalize correlated embeddings generated when finetuning BERT models, leading to a reduction in estimation errors in the speech-based prediction of cognitive impairment [33]. The original VICReg publication also highlighted the efficacy of VICReg loss, demonstrating that it could outperform both full-supervision and self-supervision with a Barlow-Twins loss on a general audio classification task, specifically on the ESC-50 dataset [9, 34]. These

studies provide evidence that VICReg loss, initially designed for image-based tasks, can be successfully adapted to other modalities, such as audio and sensor data, offering compelling performance in various applications.

2
Theory

Artificial Neural Networks (ANN) have become an integral part of modern society, with applications that span multiple industries and academic disciplines ranging from automotive to biomedical research. The following sections briefly introduce the theoretical foundations of ANNs and subsequent modern adaptations of the ANN framework used throughout this study. Moreover, we introduce Convolutional Neural Networks (CNNs) and the network architecture SampleCNN used throughout our experiments. Finally, we introduce the Self-Supervised Learning (SSL) paradigm and give a theoretical introduction to the NT-Xent and VICReg loss functions.

2.1 Neural networks

Artificial Neural networks are computing systems capable of modeling non-linear problems. Mathematically, ANN:s are function approximators: given a function $f : \mathbf{x} \mapsto \mathbf{y}$, an ANN can be written as $\hat{f} : \mathbf{x} \mapsto \hat{\mathbf{y}}$, taking input features $\mathbf{x}$ and mapping them to an approximation, $\hat{\mathbf{y}}$, of a target output $\mathbf{y}$. The mapping function $\hat{f}$ is constructed by iteratively minimizing a *loss function* $\mathcal{L}$ in a process referred to as *model training*.

The name *neural network* stems from an analogy to the human brain, as the design of neural networks was originally loosely inspired by neuroscience [35]. The word *network* illustrates that ANNs are composed of multiple interconnected functions. The most common neural network architectures, feed-forward neural networks, are composed of an acyclic chain of functions $f(\mathbf{x}) = f^{(L)}(\dots f^{(2)}(f^{(1)}(\mathbf{x})))$, where the length of the function chain, L, gives the *depth* of the model, id est the number of layers of the network [36]. Each function $f^{(i)}$ is typically vector-valued, where each element represents the current state, or *activation*, of the so-called *neurons* in that layer of the chain. The first layer, $f^{(1)}(\mathbf{x})$, is called the *input layer* since each element in this layer consists of one of the variables, or *features*, passed to the neural network as input. Similarly, the final layer, $f^{(L)}$, is called the *output layer* of the model. The intermediate layers, $i = 2, \dots, L-1$, are called *hidden layers* since they are only responsible for computation and are typically not accessed externally. The hidden layers are commonly denoted by $\mathbf{h}^{(\ell)}$, where the superscript ℓ expresses the layer index. In order to simplify formalism, we can express $\mathbf{h}^{(0)} = \mathbf{x}$ and $\mathbf{h}^{(L)} = \hat{\mathbf{y}}$.

One of the earliest forms of neural network architectures is the so-called *Multilayer Perceptron* (MLP) [36], sometimes referred to as a fully connected neural network. In

an MLP, each neuron is connected to every neuron in the preceding and subsequent layer. Typically, a *bias* vector $\mathbf{b}$ is added to the hidden layers and output layer, which biases the model to a certain value in the absence of valuable input [36]. A simple[1] MLP is illustrated in Figure 2.1, consisting of an input $\mathbf{x}$, one hidden layer $\mathbf{h}^1$, an output layer $\mathbf{y}$, and two bias vectors $\mathbf{b}_1$ and $\mathbf{b}_2$.

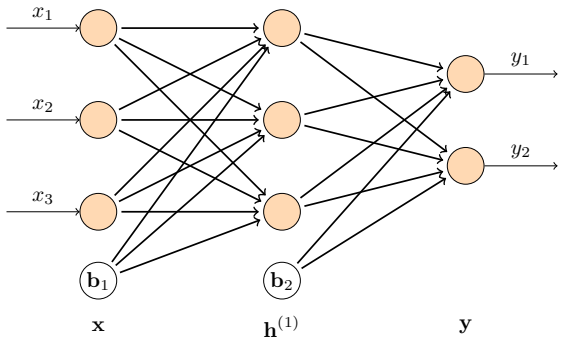

Figure 2.1: Diagram of a simple MLP consisting of an input layer $\mathbf{x}$, a hidden layer $\mathbf{h}^{(1)}$, an output layer $\mathbf{y}$, and two bias vectors $\mathbf{b}_1$ and $\mathbf{b}_2$. The interconnected arrows between the orange neurons represent the values of the associated weight matrices $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, and $\mathbf{W}^{(3)}$, respectively.

i

2.1.1 Forward propagation

Forward propagation is the process by which the input data, $\mathbf{x}$, is propagated through the neural network to generate an output prediction. In the first step of forward propagation, the input data is fed into the first layer of neurons, and each neuron in that layer computes a weighted sum of its inputs. Finally, the state of each neuron is retrieved by applying an activation function to the weighted sum. This value is then passed to the subsequent layer until the final layer is reached. In the case of MLPs, the weighted sum computed in each neuron can be represented mathematically as:

$$\mathbf{h}^{(\ell)} = \mathcal{A}\Big(\mathbf{W}^{(\ell)}\mathbf{h}^{(\ell-1)} + \mathbf{b}_\ell\Big), \tag{2.1}$$

where $\mathbf{W}^{(\ell)}$ is the ℓth column of the *weight matrix*, containing the weight of the connections between the neurons in layer $\ell - 1$ and neurons in layer ℓ, and $\mathbf{b}_\ell$ is the bias vector applied to layer ℓ. $\mathbf{W}$ and $\mathbf{b}$ are trainable parameters in a process called *backpropagation*, described in Section 2.1.3.

The function $\mathcal{A}$ acts on each element of the computation and is called *activation function*. Activation functions are crucial in designing neural network models as they introduce non-linearities to the neural network, allowing the network to model complex, non-linear functions. Consequently, the activation functions decide what information propagates to the subsequent layer and how trainable parameters are

[1] Simple in relative size and width compared to those examined in this thesis.

updated in the training phase. In recent years, the most common activation function used for this purpose is the *Rectified Linear Unit* (ReLU) [35], which is given by:

$$\text{ReLU} : \mathbf{x} \longmapsto \max(0, \mathbf{x}). \tag{2.2}$$

Consequently, this is the default activation function used throughout this report unless otherwise specified.
Another common choice for activation function is the *Sigmoid function* given by:

$$\sigma : \mathbf{x} \longmapsto \frac{1}{1 + e^{-\mathbf{x}}},$$

Which has the attractive characteristic of mapping the input values to the range [0, 1], making the sigmoid function suitable for modeling probabilities. The sigmoid function is extended to the multiclass problem setting using the following formulation:

$$\sigma : (\mathbf{x})_i \longmapsto \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}, \tag{2.3}$$

where K represents the number of classes and $\mathbf{x} = (x_1, x_2, \ldots, x_K) \in \mathbb{R}^K$. This multiclass adaptation of the sigmoid function is called the *Softmax* activation function.

2.1.2 Loss functions

The process of training neural networks refers to optimizing a set of *learnable parameters* with respect to some loss function $\mathcal{L}$. In the context of MLPs, this refers to optimizing the weights $\mathbf{W}^{(\ell)}$ and biases $\mathbf{b}^{(\ell)}$ over a given set, or *batch*, of training data such that $\mathcal{L}$ is minimized. Consequently, the loss function acts as a measurement of how well the output of a neural network $\hat{\mathbf{y}}$ corresponds to the correct labels in the training data $\mathbf{y}$.

A multitude of loss functions exists, each suitable for specific tasks a neural network is designed to address. In the context of regression problems, where the predicted output $\hat{\mathbf{y}}$ follows a continuous distribution, the most common choice of the loss function, $\mathcal{L}$, is *Mean Squared Error* (MSE), defined mathematically as:

$$\text{MSE}(\mathbf{y}, \hat{\mathbf{y}}) \equiv \frac{1}{n} \left\| \mathbf{y} - \hat{\mathbf{y}} \right\|_2^2 \tag{2.4}$$

Where n is the width of the output layer $\hat{\mathbf{y}}$. Another common task in machine learning is *classification*, i.e., determining if some given data falls within a predefined set of categories. For classification tasks, the most common choice of the loss function is *Cross Entropy Loss*, $\mathcal{L}_{CE}$, defined as follows:

$$\mathcal{L}_{CE} = -\sum_{i=1}^{n} t_i \log_2(p_i), \tag{2.5}$$

where t_i is the truth label and p_i is the Softmax probability for the i^{th} class, calculated using equation (2.3).

2.1.3 Backpropagation

Backpropagation is the process by which a neural network updates its weights and biases to minimize its loss function. As stated in equation (2.1), the output layer of a neural network depends on the parameters of each subsequent layer of the network. Thus, optimization of the loss function is achieved by calculating the gradient of the loss function, $\mathcal{L}$, with respect to the learnable parameters and then updating the parameters as follows:

$$\theta \longmapsto \theta - \eta \nabla_\theta \mathcal{L}.$$

Here, θ represents any learnable parameter, and η is a hyperparameter called *learning rate*, which controls the step size of the gradient descent. This procedure generalizes beyond MLPs and is thus used in most neural network training.

While it is possible to calculate the loss function gradients, $\nabla_\theta \mathcal{L}$, analytically, the calculations are computationally expensive due to the large number of parameters used in neural networks, the need to calculate the loss and its derivatives for each of these parameters individually, and the high complexity associated with deep network architectures. The procedure would require a separate forward and backward pass through the network for each parameter to compute its associated gradient, which becomes infeasible when dealing with networks containing millions or even billions of parameters. The backpropagation algorithm instead calculates the gradients one layer at a time, starting from the last layer. This way, the backpropagation algorithm can efficiently avoid redundant terms in the chain rule. The computational complexity can be further reduced by randomly dividing the training data into smaller batches and estimating the gradient using the average loss value across each batch in a process called *stochastic gradient descent* [37].

2.1.4 Batch normalization

Batch normalization (BN) is a technique introduced by Ioffe and Szegedy [38] that addresses a problem known as internal covariate shift, where the input data distribution changes during training, hindering the model's ability to learn an optimal representation. BN normalizes the activations of each layer, improving generalization and convergence.

The central concept of BN is to standardize input features of a mini-batch during training, using the mean and variance of the activations across the batch. The normalized activations are then scaled and shifted by learnable parameters, enabling the model to learn an appropriate scale and mean. For a mini-batch $\mathcal{B}$ of size m, and input feature vector $\mathbf{x}^{(k)}$, the BN operation is as follows:

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{k=1}^{m}\mathbf{x}^{(k)}, \tag{2.6}$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{k=1}^{m}(\mathbf{x}^{(k)} - \mu_{\mathcal{B}})^2, \tag{2.7}$$

$$\hat{\mathbf{x}}^{(k)} = \frac{\mathbf{x}^{(k)} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}, \tag{2.8}$$

$$\mathbf{y}^{(k)} = \gamma\hat{\mathbf{x}}^{(k)} + \beta, \tag{2.9}$$

where $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}^2$ denote the mini-batch mean and variance, respectively. The normalized input feature vector $\hat{\mathbf{x}}^{(k)}$ is obtained by subtracting the mini-batch mean and dividing by the standard deviation, which is the square root of the mini-batch variance. A small constant $\epsilon > 0$ is added to the variance for numerical stability. The learnable parameters γ and β are the scale and shift factors, respectively, and $\mathbf{y}^{(k)}$ denotes the output of the batch normalization operation.

2.2 Convolutional neural networks

Convolutional Neural Networks (CNNs) represent a class of deep learning architectures predominantly employed for image and signal processing tasks [39, 40]. These networks exhibit significant efficacy in discerning features within data with grid-like topologies, such as images, while maintaining a substantially reduced parameter count compared to traditional fully connected networks. In this section, we describe the key concepts of CNNs, covering sparse interactions, parameter sharing, and equivariant representations. Lastly, we introduce the SampleCNN architecture, which is leveraged as a backbone architecture for the experiments conducted in this thesis.

2.2.1 Sparse interactions

CNNs exploit so-called *sparse interactions* by restricting connections to a small subset of neurons from the preceding layer. This is achieved by employing a kernel (also known as a filter), K, of dimensions $k \times C_{in}$, where k denotes the kernel size, and C_{in} represents the number of input channels—the kernel slides across the data, applying a convolution operation at each position. The size of the step taken by the kernel while propagating the data is known as the *stride*. Given raw audio data as input, with dimensions $L \times C$, where L and C represent the length and number of channels, respectively. In a fully connected network, each neuron in the subsequent layer would be connected to all the neurons in the preceding layer, leading to a total of $(L_{in} \times C_{in}) \times (L_{out} \times C_{out})$ connections. Adopting a kernel, as opposed to a fully connected layer, the number of connections is reduced to $(k \times C_{in}) \times (L_{out} \times C_{out})$, which significantly reduces runtime since $k << L_{in}$. Additionally, CNNs often use pooling layers as intermediate steps between convolutions, which reduces output

dimensionality while preserving useful information. An example of a pooling layer is the *max-pooling* layer, which only outputs the largest of its pool of input.

2.2.2 Parameter sharing

In convolutional layers, each neuron applies an identical convolutional filter or kernel to its respective local receptive field, thereby sharing filter parameters throughout the entire layer. This technique curtails the number of learnable parameters and serves to attenuate overfitting. Furthermore, this shared parameter set enables the network to detect features irrespective of their spatial positioning within the input, engendering a translation-invariant representation of the data.

2.2.3 Equivarant representation

The notion of equivariant representations refers to the propensity of a neural network's output to undergo predictable transformations in response to specific input transformations. In the context of CNNs, this signifies that when the input undergoes a transformation, such as spatial translation, the output adapts correspondingly while preserving the learned spatial relationships. This characteristic empowers CNNs to recognize and process patterns independent of their position within the input, enhancing their robustness to spatial variations in the data.

2.2.4 SampleCNN

SampleCNN [15] is an end-to-end CNN specifically designed to process raw audio data in Music Information Retrieval (MIR) tasks, such as music auto-tagging. Its main innovation lies in the transition from frame-level to sample-level processing, allowing the model to operate directly with raw waveform data, as opposed to relying on precomputed spectrogram-based representations, which are a common practice in many MIR tasks.

The architecture of SampleCNN is tailored to extract more granular information from audio data, addressing critical challenges like log-scale amplitude compression and phase invariance [15]. This is achieved by implementing a stride at the sample level in the first convolutional layer, which tackles the phase-invariance issue, followed by repeated pairs of convolutional and max-pooling layers.

The SampleCNN architecture was established empirically through a hyperparameter search focusing on architecture depth, filter length, and stride. The best result was achieved using their so-called 3^9 architecture, featuring 9 max-pooling layers with a stride and filter length of 3, respectively. The final architecture comprises $1 + 9 + 1$ convolutional layers and 9 max-pooling layers, reducing the input dimension 3^{9+1} times from $59,049$ to 1×512. Refer to Table 2.1 for a detailed description of the SampleCNN architecture.

Table 2.1: SampleCNN architecture. In this table, Conv x-y denotes a Convolutional neural layer with stride x and y output channels. maxpool x denotes a max-pooling layer of size x.

3^9-SampleCNN Model			
59,049 Samples (2678 ms) as Input			
Layer	**Stride**	**Output**	**of Params**
Conv 3-128	3	$19,683 \times 128$	512
Conv 3-128 maxpool 3	1 3	$19,683 \times 128$ $6,561 \times 128$	49,280
Conv 3-128 maxpool 3	1 3	$6,561 \times 128$ $2,187 \times 128$	98,560
Conv 3-256 maxpool 3	1 3	$2,187 \times 256$ 729×256	196,864
Conv 3-256 maxpool 3	1 3	729×256 243×256	196,864
Conv 3-256 maxpool 3	1 3	243×256 81×256	196,864
Conv 3-256 maxpool 3	1 3	81×256 27×256	196,864
Conv 3-256 maxpool 3	1 3	27×256 9×256	196,864
Conv 3-512 maxpool 3	1 3	9×512 3×512	393,728
Conv 3-512 maxpool 3	1 3	3×512 1×512	786,944
Conv 3-512 dropout 0.5	1 -	1×512 1×512	262,656
Total params			2.46×10^6

2.3 Self-supervised learning

Self-supervised learning (SSL) is a subfield of machine learning that focuses on learning valuable representations of data without the need for explicit supervision, *i.e.*, without labeled data. Unlike supervised learning, which relies on labeled data, SSL relies on designing a pretext task that can be solved using the data's inherent structure. Commonly, the pretext task is designed to predict a partially corrupted version of the input or to compare contrasted pairs of inputs [2]. Minimizing a loss function in the learning process enables the model to identify and capture relevant patterns

within the input data based on the pretext task. Examples of loss functions for SSL in the audio domain include NT-Xent coined in SimCLR and triplet loss [4, 31, 15, 3].

This study focuses on SSL methods to generate a joint embedding or latent space containing a valuable representation of input data using multiple variations, or augmentations, of that data. This approach falls within the SSL paradigm known as *Joint Embedding Architectures* (JEA) [2]. These architectures process multiple, possibly augmented, views of a single input signal through *encoders*, producing low-dimensional representations of the views. Models in the JEA paradigm often rely on neural network architectures with multiple branches of encoders working in parallel. A typical case of such branches is Siamese neural networks, where two identical encoders share weights, as illustrated in Figure 2.2.

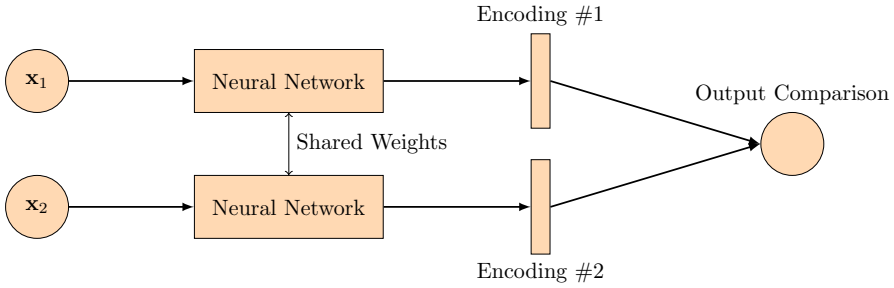

Figure 2.2: Diagram of a Siamese neural network consisting of two inputs, $\mathbf{x}_1$ and $\mathbf{x}_2$, fed to two identical neural networks with shared weights that produce two output encodings. Finally, the encodings are compared with respect to some distance metric d.

Siamese neural networks are a class of neural networks designed to learn the similarity or dissimilarity between input data pairs. These networks consist of two identical subnetworks, each with the same architecture and shared parameters, that independently process different segments or augmentations of the input data. Given an input pair (x_i, x_j), which are different temporal segments or augmentations of the same input, a siamese neural network computes the embeddings $f(x_i)$ and $f(x_j)$ using the shared parameters [41]. The embeddings are then combined to compute a similarity or dissimilarity score, typically based on the Euclidean distance. This score is then used to update the shared parameters of the Siamese network using a task-specific loss function, enforcing that the learned representation is invariant to the difference between pairs of inputs.

Self-supervised learning architectures can generally be separated into two classes, *contrastive learning* and *non-contrastive* learning. In both cases, the objective is to ensure that the representations of different signals are distinct (*i.e.*, to avoid *information collapse*) and, in JEA, to achieve closeness between related pairs of data. How this is achieved, however, differs between contrastive and non-contrastive methods as follows:

contrastive learning Avoids collapse and achieves similarity by leveraging a loss function that minimizes the distance between *positive pairs* of data while maximizing the distance between *contrastive samples*. In this setting, positive samples refer to augmented views of the same data point, and contrastive samples to data points originating from different inputs [1]. A common contrastive loss function is *NT-Xent* loss, described in Section 2.3.1.

non-contrastive learning Achieves similarity, then prevents collapse through architectural constraints or by regularization in the training objective or *energy* (*i.e.*, the representation predictive error for JEA). In JEA, information collapse is avoided by maximizing the information content of the representations [1], *e.g.*, by regularizing the covariance matrix and variance of the representations as in VICReg loss, described in Section 2.3.2.

2.3.1 NT-Xent

The NT-Xent (*Normalized Temperature-scaled Cross-Entropy*) loss is another contrastive loss function used for self-supervised learning in music track tagging. It is designed to encourage the embeddings of different augmentations of the same music track to be similar while pushing the embeddings of unrelated music tracks apart.

The NT-Xent loss is defined as follows. Let $\mathbf{x_i}$ be an input data point and $\mathbf{z_i}$ be its corresponding encoding representation obtained from a neural network. We define a *positive pair* as a pair of representations $\mathbf{z_i}$ and $\mathbf{z_j}$ that come from the same input data point $\mathbf{x_i}$, and a *negative pair* as a pair of representations $\mathbf{z_i}$ and $\mathbf{z_k}$ that come from different input data points $\mathbf{x_i}$ and $\mathbf{x_k}$.

Let $\mathcal{P}$ denote the set of all positive pairs. The NT-Xent loss for a batch of input data points is defined as follows:

$$\mathcal{L}_{NT-Xent} = -\frac{1}{N} \sum_{(i,j)\in\mathcal{P}} \log \frac{\exp(\text{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)}{\sum_{k=1}^{2N} \mathbf{1}_{[k\neq i]} \exp(\text{sim}(\mathbf{z_i}, \mathbf{z_k})/\tau)}, \tag{2.10}$$

where N is the batch size, $\text{sim}(\mathbf{z_i}, \mathbf{z_j})$ is a similarity function (*e.g.,* cosine similarity or dot product), $\mathbf{1}_{[k\neq i]} \in \{0, 1\}$ is an indicator function evaluating to 1 if $k \neq i$, and τ is a temperature parameter that controls the scale of the similarity function. The first term in the numerator represents the similarity between the positive pair, and the denominator represents the sum of similarities between the positive and negative pairs.

2.3.2 VICReg

VICReg (Variance-Invariance-Covariance Regularization) is a self-supervised method for training joint embedding architectures that aim to preserve the information content of the latent space while preventing collapse. Unlike popular contrastive methods, VICReg avoids informational collapse even with small batch sizes and does not

push dissimilar images away; this is appealing as it circumvents a common issue in contrastive learning: the unintended distancing of closely related samples as a consequence of driving dissimilar ones apart.

Further, VICReg extends beyond the confines of Siamese Networks architectures; there is no obligation for its two branches to mirror each other in terms of parameters, architecture, or input modality. This allows for employing a non-contrastive self-supervised joint-embedding method for multi-modal data, such as video and audio.

Like common practices in the contrastive domain, the method encompasses an encoder, f_θ, and a projector, h_ϕ. The encoder yields latent space representations for subsequent tasks, while the projector facilitates mapping into an embedding space to calculate the loss. The VICReg loss function constituents three terms:

1. Invariance: s is a measurement of the similarity between the input x and its augmented counterpart x'.

2. Variance: v is a regularization term that contains a trained constant standard deviation of the encoded latent space.

3. Covariance: c prevents information collapse by decorrelating non-diagonal elements in the covariances of the batch.

After pre-training, the projector is discarded, and the encoder is harnessed for downstream tasks.

In our setup, we adopt a Siamese architecture. Then the mathematical formulation of VICReg loss is as follows. Let us consider i as an instance of audio clips drawn from a dataset $\mathcal{D}$. The transformations $x = t(i)$ and $x' = t'(i)$ are obtained from a transformation distribution, denoted as $\mathcal{T}$. By feeding these transformations through the two branches of the network, we acquire the representations $y = f_\theta(x)$, $y' = f_\theta(x')$, and correspondingly, the embeddings $z = h_\phi(y)$, $z' = h_\phi(y')$. The data points are sampled in batches denoted $Z = [z_1, ..., z_n]$ and $Z' = [z'_1, ..., z'_n]$., where n corresponds to the batch size, and d signifies the dimension of the vector. z^j denotes all vectors at the j-th dimension in batch Z. The variance term is defined as:

$$v(Z) = \frac{1}{d}\sum_{j=1}^{d} \max(0, \gamma - S(z^j, \epsilon)). \tag{2.11}$$

In this equation, γ stands as the target for the standard deviation, and ϵ is a small scalar introduced to prevent numerical instability. The function S denotes a regularized standard deviation and is defined as:

$$S(x, \epsilon) = \sqrt{\mathrm{Var}(x) + \epsilon}. \tag{2.12}$$

This constraint ensures a constant standard deviation of γ along the batch dimension, mitigating informational collapse. The invariance is the mean-squared Euclid-

ian distance between pairs of vectors Z and Z':

$$s(Z, Z') = \frac{1}{n} \sum_i \left\| z_i - z'_i \right\|_2^2. \tag{2.13}$$

The covariance is defined as

$$c(Z) = \frac{1}{d} \sum_{i \neq j} [C(Z)]_{i,j}^2. \tag{2.14}$$

where C is the covariance matrix defined as follows:

$$C(Z) = \frac{1}{n-1} \sum_{i=1} (z_i - \bar{z})(z_i - \bar{z})^T, \quad \text{where} \qquad \bar{z} = \frac{1}{n} \sum_{i=1}^{n} z_i, \tag{2.15}$$

This decorrelates the elements of embeddings from each other by forcing them to be close to zero, preventing them from encoding similar information. Finally, the complete loss function is formulated as a weighted summation of the variance, invariance, and covariance terms:

$$l(Z, Z') = \lambda s(Z, Z') + \mu[v(Z) + v(Z')] + \nu[c(Z) + c(Z')]. \tag{2.16}$$

In the above equation, λ, μ, and ν function as hyperparameters that modulate the weight of the different components in the loss function.

In the original paper, VICReg was employed for computer vision tasks. To our knowledge, it has yet to be explored in the audio domain, except as a secondary analysis published in the appendix of the VICReg paper as an additional result [9]. This paper aims to explore its viability in the musical domain and further explore various hyperparameters.

3
Method

The core of this thesis lies in exploring how the VICReg loss function compares in performance to the commonly used NT-Xent loss function in the music domain. This chapter presents the procedural development of such a comparison and highlights the model selection process, hyperparameter tuning, and loss adaptation. Moreover, we describe how the proposed VICReg loss function will be compared to the traditional NT-Xent loss in the subsequent chapters and give the implementation details necessary for reproducing our results.

3.1 Architectures

Before describing the experiments conducted throughout this thesis, we introduce the VICReg and CLMR architectures used for our self-supervised representation learning. Following the training and generation of the SSL latent space, the models were evaluated on a downstream task using small MLP models. This section will present the architecture used in each of the three models. The models were implemented using the *PyTorch* framework in the Python programming language. Further implementation details for each model are outlined in Section 3.3.

3.1.1 VICReg

For comparability, we employ a siamese network architecture heavily inspired by CLMR [4] to generate a latent space for audio representations. This setup, denoted VICReg, is illustrated in Figure 3.1. During training, the VICReg model takes a batch of raw audio clips as input and generates two augmented views of each input. Each view consists of a randomly cropped segment of length $59,049$, corresponding to approximately 2.7s at a $22,050$ Hz sample rate, from the input audio, which is transformed through a random sample of audio augmentations $t \sim \mathcal{T}$ according to Section 3.3.1.

Once the augmented views of the input audio are generated, each view is passed through an encoder f_θ, also denoted backbone, consisting of a SampleCNN [15] that outputs feature maps Y and Y' for the respective views. A detailed description of SampleCNN architecture is contained in Section 2.2.4. Each view is then processed through a series of fully connected neural networks h_ϕ, referred to as projection head.

Following the evaluation performed in the CLMR and VICReg studies, we perform

our downstream evaluations using the encoded feature maps Y and Y' instead of the projector output Z and Z', as this has been shown to improve performance [4, 9]. The invariance loss, $s(Z)$, minimizes the Euclidean distance between the projections, ensuring that the representation is invariant to input augmentations according to Equation (2.13). Concurrently, the variance loss, $v(Z)$, is applied across the entire input batch to enforce a target standard deviation γ according to Equation (2.11). The variance loss improves the numerical stability of the representation by ensuring that similar inputs do not map to the same point in the learned representation space, causing information collapse. Finally, the covariance loss term, $c(Z)$, in equation (2.14) is applied across the batch. This decorrelates the elements of the latent space and promotes information maximization in the learned representation.

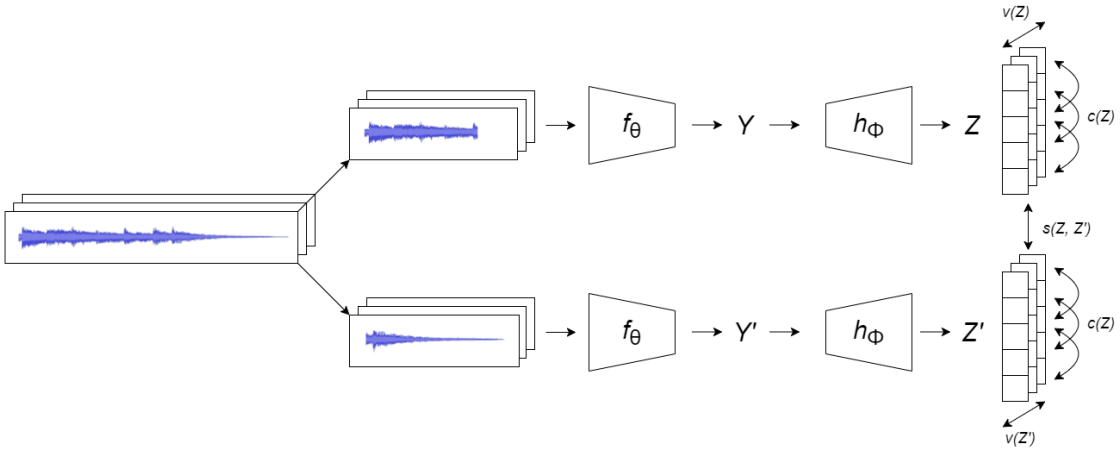

Figure 3.1: Illustration of the VICReg network architecture used for self-supervised representation learning with raw audio inputs. Given a batch of raw audio clips, our model generates two cropped and augmented representations for each audio clip in the batch, sampled according to Section 3.3.1. Then, each audio clip is then passed through a SampleCNN encoder, f_θ, described in Section 2.2.4, and linear projection layers h_ϕ. The VICReg loss is then applied to the projection output, Z, according to Section 2.3.2.

3.1.2 CLMR

The implementation of the CLMR architecture follows the same batch, augmentation, encoder, and projector design as the VICReg model, illustrated in Figure 3.1. The difference between the two models lies in the implementation of the loss functions. Letting $Z_{1,j}$, $Z_{2,j}$ represent the projection layer outputs for the two augmented views of input audio track j, NT-Xent loss acts by minimizing the distance between $Z_{1,j}$ and $Z_{2,j}$, similar to the invariance loss used in VICReg. Information collapse is avoided by repelling representations stemming from different input tracks, *i.e.,* by maximizing the distance between $Z_{i,N}$ and $Z_{j,M}$ for $i \in \{1, 2\}$ and $N \neq M$.

3.2 Experiments

This section describes the experiments conducted in this thesis and how each experiment contributes to the purpose and aim of the study. The results of our experiments on model convergence, representation size, and transfer learning are reported in Section 4. Additional experiments were performed for batch size tuning of VICReg and projector configurations for VICReg and CLMR, respectively. The additional results are listed in the Appendix B.

3.2.1 Convergence

We compare the performance of our VICReg model with the benchmark CLMR model at various stages in the training process to evaluate the rate at which the representations learn valuable information. Each model representation is trained on the same subset of the FMA dataset for 300, 1,000, 3,000, 5,000, and 8,000 epochs, respectively. The models are then evaluated in their ability to perform multi-label classification on the top 50 tags of the MagnaTagATune dataset [18], following standard conventions in the music domain [15, 18, 42].

3.2.2 Representation Layer

In this experiment, we perform a comparative analysis of the VICReg and benchmark CLMR models conducted over various dimensions for the learned representations. This result evaluates the models' capacity to retain information across various levels of dimensionality reduction. It also assesses the practical usability of the models in contexts that require compact representations, such as audio similarity, which necessitates the computation of a distance metric across each dimension. The VICReg and CLMR backbones are trained with representations of sizes 32, 64, 128, 256, and 512. After pre-training, the performance for each representation size is evaluated using the same MagnaTagATune procedure as outlined in Section 3.2.1.

3.2.3 Transfer Learning

We test the out-of-domain generalizability of the representations learned by our VICReg model by pre-training it on two distinct datasets and comparing the resulting representations with those generated by other popular SSL models in the audio domain. Following standard SSL conventions, we evaluate the VICReg representations using linear models, reflecting the relevant classes' linear separability under the learned representation [43, 44, 4]. After pre-training, the weights of our representation are fixed, thereby excluding any encoder fine-tuning. We then conduct linear evaluations on the top 50 tags of the MagnaTagATune dataset, employing a single-layer logistic classifier and double-layer MLP for comparability with the CLMR experiments.

3.2.4 Extended Experiments

This thesis primarily investigates the comparative performance of VICReg and CLMR. However, an assortment of additional experiments was conducted that, while not directly linked to our central research question, revealed intriguing patterns and insights about VICReg. These insights have been compiled into an 'Additional Results' in Appendix B. Firstly, we evaluated how the batch size affects the models' performance. Batch size, often a practical limitation dependent on available computational resources, significantly influences model training dynamics and the quality of the learned representations. For VICReg, in particular, its Variance term is computed along the batch dimension, so larger batch sizes include more variables in its computation and may alter its representation. Secondly, we explored the impact of different projectors on model performance. The choice of the projector, mapping the raw representation to a higher or lower-dimensional space, can substantially influence the effectiveness of the learning process. In this study, we performed a comparative analysis of the projectors used in the original VICReg and CLMR implementations [9, 4]. Lastly, we conducted a comprehensive grid search across batch and representation sizes for VICReg. The findings from this grid search further illuminate how these two parameters interact to affect overall model performance.

3.3 Implementation details

This section aims to provide the necessary details to reproduce our results. In Subsection 3.3.1, we present and describe all data augmentations used. Augmentation plays a crucial role in the success of self-supervised learning, enhancing the ability of our models to generalize by exposing them to more significant data variance. Subsection 3.3.2 details the software and hardware used. The 3.1 Subsection outlines the specific parameters for training our models. Lastly, Subsection 3.3.4 details the specific optimizer and learning rate schedules used while training our models.

3.3.1 Data augmentations

To ensure a fair comparison between our proposed VICReg model and the baseline CLMR model, we adopt the same audio augmentations as described in CLMR [4] in both settings. Audio augmentations are essential for improving the robustness and generalization capabilities of self-supervised learning models[10] by encouraging them to learn invariant features from raw audio waveforms. Moreover, randomly sampling augmentations from a distribution $t \sim \mathcal{T}$ effectively alters the size of the training data, allowing the model to train on a larger set of unseen samples. The distribution of augmentations $\mathcal{T}$ and the order in which they are applied are as follows:

1. **RandomCrop:** A random segment of length $59,049$ is selected from a full audio clip with a $22,050$ Hz sampling rate without removing silence (e.g., the intro or outro of a song). The independently chosen segments of the input $\mathbf{x}$ could overlap or be very disjoint, allowing the models to infer both local and global structures.
2. **PolarityInversion:** Inverts the audio signal by multiplying the amplitude by -1. Applied with a probability of 80 %.

3. **Noise:** Applies additive white Gaussian noise using a signal-to-noise ratio of 80 decibels. Applied with a probability of 1 %.

4. **Gain:** reduces the audio signal between 0 and -6 decibels. Applied with a probability of 30 %.

5. **HighLowPass:** Applies a high-pass or low-pass filter using cut-off frequencies drawn from a uniform distribution on $[200, 1200]$ or $[2200, 4000]$ Hz, respectively. The filter is determined by a Bernoulli trial with a probability of 50 % for each filter. Applied with a probability of 80 %.

6. **Audio:** adds the audio signal onto itself with a delay. The applied signal is added with a 0.5 factor in volume, and the delay is drawn randomly from the range $[200, 500]$ ms with 50 ms increments. Applied with a probability of 30 %.

7. **PitchShift:** The audio signal is pitch-shifted with a transposition interval drawn from a uniform distribution of semitones between $[-5, 5]$. Applied with a probability of 60 %.

8. **Reverb:** Applied with room size, reverberation, and damping factor, draw uniformly on $[0, 100]$. Applied with a probability of 60 %.

For training downstream tasks, the only augmentation used was **RandomCrop**. For the test set evaluation, no augmentations were used. The audio file was instead split and concatenated into a batch; the prediction was averaged over the splits.

3.3.2 Framework and Hardware

All experiments were run using Python 3.8 and PyTorch Lightning 2.0. All backbone training ran on TpuV3-8 virtual machines with 96 CPU cores, 335 GB RAM, and a total of 128 GB on-chip memory. Training VICReg with SampleCNN as its backbone on FMA for $1,000$ epochs took approximately 24 hours. Our longest experiment, using $8,000$ epochs, therefore took approximately 8 days.

3.3.3 Hyperparameters

All self-supervised methods used a Siamese network with SampleCNN as a backbone with a projector head on top of it. Backbones were trained on FreeMusicArchive and GTZAN. VICReg used a projector with three layers, each with a size of 8196. The resulting projector then has the dimensions $512 - 8196 - 8196 - 8196$ for VICReg and $512 - 512 - 128$ for CLMR. We used the same variance, invariance, and covariance coefficients as in VICReg, *id est* 25, 25, 1, respectively. As shown in Section B.1, we found that a batch size of 768 provided the best result for VICReg using a representation embedding of size 512. We used LARS optimizer as described in 3.3.4. All backbone training hyperparameters can be found in Table 3.1.

In our reimplementation of CLMR, we used the same hyperparameters as stated in the original paper, with the exception of batch size. In our setting with 8 TPU cores, a batch size of 256 proved to yield significantly better results. This was also the maximum size that could fit in the on-chip memory of our TPU virtual machines.

When evaluating our model on downstream tasks, we train a one-layer or two-layer MLP atop the representations learned from our backbone pre-training. During the retraining phase, we freeze the weights of our backbone model and hence only train the small MLP models atop our representations. The two-layer MLP uses a hidden layer with the same size as the representation size of the backbone, and the hidden layer applies the activation function ReLU. For example, a representation size of 512 and a 50-label downstream-task dataset would yield the following MLP dimensions: $512 - 512 - 50$.

3.3.4 Optimizers

In our implementation, we employed the Layer-wise Adaptive Rate Scaling (LARS) optimizer [45] with hyperparameters consistent with those utilized in VICReg [9]. This encompassed a weight decay of 10^{-6} and a learning rate calculated as $lr = batch\_size/256 \times base\_lr$, where the base learning rate was set to 0.2. Unlike optimizers such as Adam [46], which adjust according to individual weights, LARS notably adapts its learning rate on a layer-by-layer basis.

For the CLMR model, we adhered to the settings detailed in the original paper [4]. Specifically, an Adam optimizer [46], a learning rate of 0.0003, and $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Table 3.1: Key hyperparameters utilized in our VICReg implementation.

Hyperparameter	VICReg
Epochs	$1,000$
Precision	bfloat-mixed
Sample rate	$22,050$
Batch size	7.68
Backbone architecture	SampleCNN [15]
Optimizer	LARS [45]
Base learning rate	0.2
sim_coeff	25
std_coeff	25
cov_coeff	1
Weight decay	10^{-6}

When training on downstream tasks, we followed the settings in CLMR [4], which utilized an Adam optimizer with a learning rate of 0.001 and a weight decay of 10^{-6}. We used the *PyTorch* built-in learning rate scheduler *ReduceLROnPlateau* where the learning rate was multiplied by 0.1 whenever the loss did not decrease within five epochs.

3.4 Data

Following the transfer learning experiments performed in CLMR [4], three publically available datasets are used to test the out-of-domain generability of the learned representations in this thesis. The representations (embeddings) are trained on Free Music Archive (FMA) [16], and GTZAN [17], respectively. A shallow neural network is then trained on top of the embeddings using the MagnaTagATune [18] dataset, which consists of audio tags, to evaluate the generalizability of the embedding to downstream tasks.

3.4.1 GTZAN

The GTZAN dataset [17], introduced by George Tzanetakis and Perry Cook, is one of the most widely-used datasets for music genre classification tasks [47]. It consists of $1,000$ audio tracks, with each track being 30 seconds in length with a $22,050$ hz sample rate, evenly distributed across ten genres: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock. Each genre contains 100 samples, providing a balanced dataset for classification experiments.

Despite some known issues, such as mislabeling and duplicate tracks [47], the GTZAN dataset remains a popular benchmark for music genre classification due to its simplicity and accessibility. Following the implementation of CLMR [4], we use a fault-filtered GTZAN dataset consisting of 930 song fragments where known issues have been removed.

3.4.2 FreeMusicArchive

The Free Music Archive (FMA) dataset [16] is a large-scale collection of music from various genres sampled by the Free Music Archive platform. It contains over 100,000 tracks, spanning 161 genres, and is distributed under Creative Commons licenses, which makes it suitable for research purposes.

The FMA dataset is divided into three subsets based on the number of tracks: small (8,000 tracks), medium (25,000 tracks), and large (106,574 tracks). Each dataset provides a diverse source of audio samples commonly used in analysis tasks, such as genre classification, tag prediction, and recommendation systems. Its large-scale nature and extensive metadata make it particularly useful for training and evaluating deep learning models, such as self-supervised representation learning. In this study, we train our backbones using the FMA medium dataset.

3.4.3 MagnaTagATune

The MagnaTagATune (MTAT) dataset [18] was compiled by crowdsourcing musical tags through a game called "TagATune," which utilized music from the Magnatune label. It consists of 25,863 music clips, each 29 seconds long, and focuses on multi-label tagging tasks, with each clip annotated by multiple users participating in the game.

The original dataset contains 188 unique tags, some of which are synonymous (e.g., "female," "woman," "no vocal," "no voice") and includes tracks that do not have any labels. For this study, we follow the same train/test split as previous work [42, 4] and use only the top 50 tags to maintain consistency and comparability with other studies [4, 48, 15].

MagnaTagATune is valuable for music information retrieval tasks, particularly multi-label classification and tag prediction. The dataset's size, diverse set of user-generated tags, and the top-50 tags subset provide valuable ground truth for developing and evaluating machine learning algorithms in the context of music analysis.

3.5 Evaluation Metrics

In order to evaluate our self-supervised representation learning models on the MagnaTagATune dataset and to compare our work to CLMR [4], and align with previous studies [42, 15], we employ two multi-label evaluation metrics: *Receiver Operating Characteristic Area Under the Curve* (ROC-AUC) and *Precision-Recall Area Under the Curve* (PR-AUC). The ROC-AUC and PR-AUC metrics are used to assess model performance tag-wise (by column) and clip-wise (by row). We follow the conventions used in previous works and multiply all ROC-AUC and PR-AUC results by a factor of 100 for improved readability.

3.5.1 ROC-AUC

ROC-AUC is a widely used evaluation metric for binary classification problems. It measures the area under the Receiver Operating Characteristic (ROC) curve, which summarizes the trade-off between the *True Positive Rate* (TPR) and the *False Positive Rate* (FPR) at various classification thresholds. Mathematically, TPR and FPR can be expressed as:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \ \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}},$$

where TP, FP, FN, and TN denote true positives, false positives, false negatives, and true negatives, respectively. A ROC-AUC score of 100 indicates a perfect classifier, while a score of 50 signifies a random classifier. However, ROC-AUC can produce over-optimistic scores when dealing with imbalanced datasets. Hence, we also employ the PR-AUC metric.

3.5.2 PR-AUC

PR-AUC is an evaluation metric that measures the area under the *Precision-Recall* (PR) curve. Precision (also known as Positive Predictive Value) quantifies the proportion of true-positive predictions among all positive predictions. Recall (also known as Sensitivity) measures the proportion of true positives among all relevant instances. Precision and Recall can be defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \ \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

The PR-AUC metric is beneficial in cases of class imbalance, as it focuses on the classifier's performance for the positive class.

Both ROC-AUC and PR-AUC metrics are evaluated globally for the whole dataset. For the tag metric, we measure the retrieval performance on the tag dimension (column-wise), while for the clip metric, we measure the performance on the clip dimension (row-wise). The tag retrieval performance reflects how well the model retrieves all music fragments (clips) given the tags. In contrast, the clip retrieval performance indicates how well the model retrieves all tags given a clip.

4
Results

In this chapter, we give a performance evaluation of the VICReg [9] loss function applied in the musical domain and a comparison with CLMR [4]. This is achieved by comparing the ROC-AUC and PR-AUC performance of VICReg and NT-Xent in a series of experiments involving model convergence and transferability across various downstream tasks and datasets. In the following sections, ROC-AUC and PR-AUC scores given within parentheses signify results achieved with a two-layer MLP classifier, and scores without parentheses signify results achieved using a linear classifier.

4.1 Convergence

In Figure 4.1, we report the tagging performance of VICReg and CLMR when pre-trained on an incremental number of epochs on the FMA dataset. We evaluate both methods on MTAT using PR-AUC and ROC-AUC using backbones trained for 300, $1,000$, $3,000$, $5,000$, and $8,000$ epochs, respectively. The exact values are found in Table A.1.

For the CLMR model, the ROC-AUC score reaches a peak of 87.52 (88.12) when trained for 300 epochs. However, it declines and levels at 86.22 (87.48) after $8,000$ training epochs. The PR-AUC score for the same model similarly drops from its peak of 33.40 (33.83) to 31.33 (32.73) as training epochs increase.

Conversely, the VICReg model displays an incremental improvement throughout the training epochs. Starting from a ROC-AUC score of 87.08 (88.19) at 300 epochs, the model gradually improves this score to 87.93 (89.14) by the end of $8,000$ epochs. This trend is also reflected in the PR-AUC score, which rises from 32.24 (33.55) to 34.64 (35.47) during the same period.

These results depict an opposite trend between the VICReg and CLMR models, where VICReg continually improves with longer pre-training, while CLMR appears to reach a performance peak early and then gradually declines.

4.2 Representation Layer

We conduct experiments where we alter the size of the representation layer and evaluate the performance of both VICReg and CLMR models using ROC-AUC and

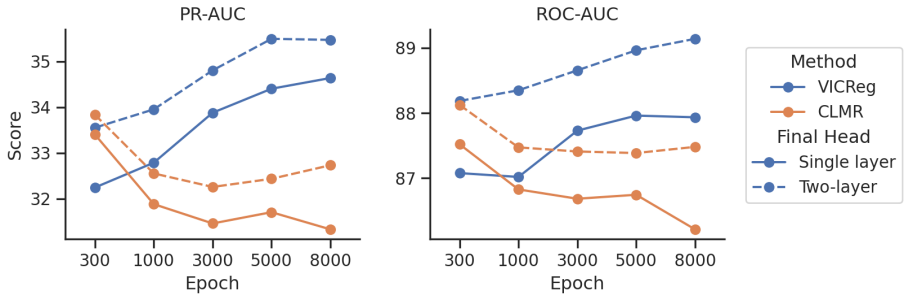

Figure 4.1: VICReg and CLMR convergence when they are pre-trained for 300, 1,000, 3,000, 5,000, 8,000 epochs on the FMA dataset. They are evaluated on MTAT using ROC-AUC and PR-AUC using both a single-layer and two-layer final head.

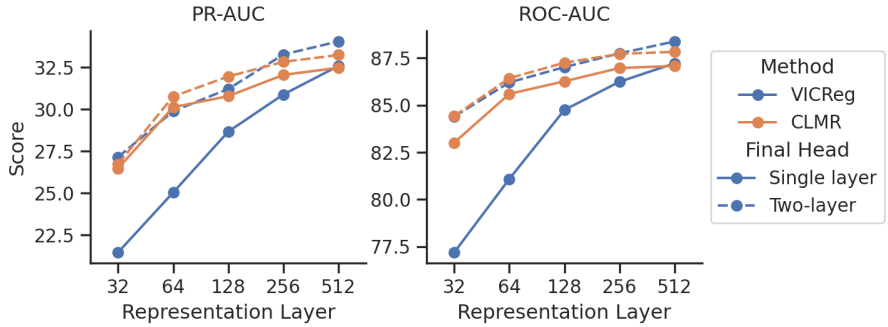

Figure 4.2: Performance of VICReg and CLMR architectures as a function of representation layer size. The table presents ROC-AUC and PR-AUC scores for representation layer sizes 32, 64, 128, 256, and 512. The scores achieved with a single-layer final head are represented as a line, and the scores achieved using a two-layer final head are shown in a dashed line. All backbones are trained for 1,000 epochs.

PR-AUC. The result can be found in Figure 4.2. The exact values are in Table A.2.

Figure 4.2 demonstrates a notable disparity between the performance of VICReg and CLMR architectures, particularly at lower representation layer sizes. For sizes 32, 64, and 128, CLMR outperforms VICReg in both ROC-AUC and PR-AUC scores. This suggests that CLMR is more adept at finding linearly separable representations in lower-dimensional spaces, indicating more efficient learning in these conditions.

However, as the representation layer size increases to 256 and 512, VICReg closes the gap, eventually surpassing CLMR in both ROC-AUC and PR-AUC scores. This shift suggests that VICReg benefits more from increased representational capacity, allowing it to capture more complex patterns and relationships in the data.

4.3 Transfer learning

Table 4.1 presents the ROC-AUC and PR-AUC on the MTAT dataset for several self-supervised methods. These results are distinguished based on the dataset utilized for pre-training. The symbol (*) denotes a supervised network. Additional information about the choice of batch sizes is available in Section 5.1.

Our proposed method, VICReg, was pre-trained for 8,000 epochs on both the GTZAN and FMA datasets while only pre-training for 3,000 epochs on the MTAT dataset. VICReg produced ROC-AUC scores of 87.86(88.87) on MTAT, 80.56(81.33) on GTZAN, and 87.78(89.15) on FMA. Meanwhile, the PR-AUC results were 34.84(35.28) for MTAT, 23.26(23.32) for GTZAN, and 34.71(35.85) for FMA.

CLMR (r) underwent training for 300 epochs for the FMA dataset and 3,000 epochs for the GTZAN dataset. Its ROC-AUC scores were 87.52(88.12) on FMA and 83.47(85.02) on GTZAN. In terms of PR-AUC, CLMR (r) achieved 33.40(33.83) on FMA and 27.20(28.89) on GTZAN.

In the context of supervised methods, MusiCNN and SampleCNN* showcased different architectures, with MusicCNN* employing a mel-spectrogram-based structure similar to PEMR, while SampleCNN* used a waveform-based architecture. Their ROC-AUC scores were 89.1 and 88.6, respectively, and the PR-AUC scores were 34.9 and 34.4.

Lastly, the CLMR method, pre-trained on the MSD dataset, demonstrated a ROC-AUC of 88.7(89.3) and a PR-AUC of 35.6(36.0).

Table 4.1: ROC-AUC and PR-AUC on MTAT for various self-supervised methods. Datasets for pre-training are listed in the column Dataset. VICReg was trained for 8,000 epochs on GTZAN and FMA, and 3,000 epochs on MTAT. CLMR (r) was trained for 300 and 3,000 epochs for FMA and GTZAN, respectively. See Section 5.1 for further reasoning surrounding the choice of batch sizes. The (r) signifies our reproduced results. Values in parenthesis signify a two-layer MLP as the final head. * indicates a supervised network.

Method	Dataset	Repr. Size	ROC-AUC	PR-AUC
MusicCNN* [49]	MTAT	-	89.0	34.9
SampleCNN*[49]	MTAT	-	88.6	34.4
CLMR [4]	MSD	512	**88.7** (**89.3**)	**35.6** (**36.0**)
CLMR (r)	GTZAN	512	83.47 (85.02)	27.20 (28.89)
CLMR (r)	FMA	512	87.52 (88.12)	33.40 (33.83)
VICReg (ours)	GTZAN	512	80.56 (81.33)	23.26 (23.32)
VICReg (ours)	MTAT	512	87.86 (88.87)	34.84 (35.28)
VICReg (ours)	FMA	512	87.78 (89.15)	34.71 (35.85)

5

Discussion

5.1 Information collapse in CLMR

We initially focus on CLMR's performance on the downstream classification task on MTAT, which peaks when trained for 300 epochs and then steadily diminishes until $5,000$ epochs, as seen in Figure 4.1. Counterintuitively, while CLMR's downstream performance on the MTAT dataset declines, its training loss on the FMA and its validation loss, evaluated on MTAT, continue to decrease, as seen in Figure A.1. This Is not a conventional case of overfitting since both the training and validation loss is continuously declining; rather, it seems to lose its ability to retain relevant information for the downstream task at hand.

This may be due to informational collapse, which happens when a self-supervised method loss decreases but loses information in its representation. For example, when the loss is defined by an invariance such as the ℓ_2 norm, an optimal solution is shrinking all vectors to a single point. This has previously been empirically observed in both self-supervised and unsupervised settings [50, 51, 12]. Contrastive methods aim to combat this by pushing contrastive negative pairs away from each other; intuitively, the repulsive effects leverage the collapse. It has been shown that despite this, contrastive methods still exhibit information collapse, but only partially through dimensional collapse where only collapse occurs in some dimensions[12].

While good data augmentations are essential to training useful representations[10], good data augmentations might cause collapse on smaller datasets such as FMA. Previous studies have empirically shown how dimensional collapse in contrastive methods may occur when strong augmentations along the feature dimension present a variance larger than its data distribution [12]. In the original CLMR publication, Spivjkvert et al. [4] trained the CLMR model on the significantly larger Million Song Dataset (MSD), which may naturally exhibit greater variance than FMA. In their reported results, there were no indications of decreasing performance or information collapse [4]. Further work is required to study the effects of varying datasets on the CLMR model.

As observed in Figure 4.1, VICreg does not exhibit this problem. This is likely due to its regularization terms, which are designed to encourage the model to make the most out of the entire feature space and, crucially, to maintain diverse information across different dimensions while avoiding information collapse.

The variance term in VICReg encourages the model to utilize the entire feature space uniformly by ensuring variation across the representation layers dimensions, avoiding single-point collapse. In addition, the covariance term minimizes the redundancy among different dimensions by reducing the correlation between them. This discourages the model from storing similar information across multiple dimensions and instead promotes a broader, more diverse representation. Such a design ensures that even when the model is excessively trained, it retains various information across different dimensions, mitigating the risk of informational collapse.

Based on these observations, VICReg seems to be generally less susceptible than contrastive methods such as CLMR to informational collapse, at least in settings with small data distributions.

5.2 Representation size

Interestingly, as seen in Figure 4.2, VICReg consistently underperforms compared to CLMR for representation sizes up to 256. However, at a representation size of 512, the VICReg architecture outperforms CLMR. This finding suggests that the regularization mechanisms within VICReg, which aim to distribute information evenly and reduce redundancy across dimensions, may require higher-dimensional representation spaces to manifest their effectiveness.

Additionally, we note a significant observation regarding the linear separability of the representations. It is evident from the figure that as the representation size decreases, the representations become less linearly separable, particularly for VICReg. This is reflected by the larger performance increase achieved by adding a second layer in the final head for smaller representation sizes, especially when compared to the performance increase at larger representation sizes. It could suggest that the representations learned by VICReg in lower-dimensional spaces may be more complex or non-linear and, thus, more challenging to separate using a simple linear classifier.

In light of these findings, we can argue that while CLMR might be preferable for smaller representation sizes, VICReg shows promising results when equipped with a larger representation layer, exhibiting superior performance and more effective use of higher-dimensional representation space.

5.3 Transfer learning

This research offers intriguing findings while also uncovering potential areas for further investigation. First and foremost, an unforeseen challenge arose during the training phase due to intermittent TPU crashes, resulting in different numbers of

training epochs across our experiments. This unpredictability was particularly pronounced while working with the MTAT dataset, which did not fully converge due to these complications. This situation underlines the critical need for stable computational resources when dealing with complex machine learning models and the potential requirement for optimizing the parameters specific to certain datasets.

When compared against CLMR, VICReg, as detailed in Table 4.1, yields superior results to CLMR when training on FMA within our test suite. VICReg nearly equates to the result of CLMR when trained on ten times as few samples, FMA against MSD. We have yet to see how training on VICReg on a much larger dataset would affect the performance; further investigation would prove interesting.

In contrast, CLMR outperforms VICReg on GTZAN, in conjunction with our representation experiment (Section 4.2), we believe this result might indicate that CLMR generalizes better than VICReg in low information settings, being either on small datasets such as GTZAN or using small representation sizes.

Our results show that pre-training with a SampleCNN architecture using VICReg performs better at music tagging on MTAT than training a supervised SampleCNN architecture directly on MTAT. Consequently, we confirm that training a representation using VICReg can improve performance on downstream tasks compared to its fully supervised counterpart. On the other hand, VICReg does not beat the supervised musicnn [52], but this model is much larger than SampleCNN (12 million versus 2,5 million parameters) and uses mel-spectrogram as input - which is generally known to outperform waveform-based ones[15].

5.4 Future work

5.4.1 Performance improvement

Regarding the further performance improvements of VICReg, there are several promising avenues to explore.

Firstly, one can investigate different backbone models for VICReg, especially those involving larger representation sizes. Previous studies have shown that, given enough data, larger models often improve model performance [53, 54, 55, 56]. Moreover, there is strong empirical evidence that the same holds for SimCLR [57]. Finally, it has been shown that larger representation sizes improve the performance of VICReg on downstream tasks [58]. Our results have confirmed the performance of VICReg increases when using larger representation sizes, as seen in Figure 4.2, suggesting that this also applies to the music domain. It would be interesting to explore how larger models with larger representation sizes affect the performance of VICReg on downstream tasks. Further, this would make our results more directly comparable to other works since representation sizes vary significantly across studies [9, 59].

Data augmentation strategies also hold considerable potential for enhancing the performance of SSL methods. In this study, we applied the same augmentations used in CLMR, but many SSL papers within the Music domain used different augmentations that may be worth exploring. Improved augmentations could contribute to the development of more robust and generalizable representations.

In training our VICReg model, we employed the Layer-wise Adaptive Rate Scaling (LARS) optimizer, adopting the configurations proposed in the original VICReg paper. This was despite our departure from their approach, which utilized a pre-trained backbone. Although LARS demonstrated its effectiveness in our context, we identified opportunities for further performance enhancements through nuanced hyperparameter calibration. A notable observation during training was VICReg's apparent advantage from exposure to larger learning rates over more epochs. Consequently, we identify the exploration and fine-tuning of the base learning rate and learning rate schedule within the LARS optimizer as a promising direction for future study and optimization of model performance.

5.4.2 Pre-training and evaluation dataset

In the pursuit of enhancing the performance and fortifying the findings of this work centered around VICReg, we propose two promising paths for future research:

1. **Investigating Larger Pre-training Datasets**: To further enhance the performance and generalizability of VICReg, it would be interesting to explore larger pre-training datasets. For example, the Million Song Dataset (MSD) offers a vast collection of music data from various genres and artists. It has been used in multiple studies as an SSL pre-training dataset [4, 13], including CLMR [4], see Table 4.1. Therefore, it would also enhance comparative analyses with previous works while concurrently reducing discrepancy.

2. **Integration of Additional Downstream Tasks**: The potency and transferability of the representations learned through VICReg can be further corroborated by integrating more varied downstream tasks. One promising option is to utilize the recently proposed HARES dataset by DeepMind [22], which provides 12 different multi-label audio classification tasks within various domains: environmental, speech, and music. Training on the music subset of HARES would yield valuable insights and allow for a broader comparative base with earlier works.

6
Conclusion

In summary, our work constitutes a step forward in applying SSL techniques in the audio domain, with VICReg demonstrating competitive performance compared to CLMR. We believe VICReg can achieve superior performance with more robust training. We advocate continued exploration in this direction.

Given the same model backbone and training data, VICReg compares to CLMR in terms of representation convergence rate with an opposite trend between the VICReg and CLMR models, where the former continually improves with longer pretraining, while the latter appears to reach a performance peak early and then gradually declines.

In terms of resilience to collapse, VICReg seems to be generally less susceptible than contrastive methods such as CLMR to informational collapse, at least in settings with small data distributions.

For the downstream task performance, VICReg consistently outperforms CLMR when pre-trained on FMA. The difference is especially clear when performing the evaluation with a two-layer MLP as opposed to the linear logistic classification. Conversely, when trained on GTZAN, CLMR significantly outperforms VICReg.

For different representation dimensions, our results show that VICReg improves performance when using larger representation sizes, suggesting that this also applies to the music domain.

Bibliography

[1] R. Shwartz-Ziv and Y. LeCun, "To compress or not to compress - self-supervised learning and information theory: A review," *ArXiv*, vol. abs/2304.09355, 2023.

[2] R. Balestriero, M. Ibrahim, V. Sobal, A. Morcos, S. Shekhar, T. Goldstein, F. Bordes, A. Bardes, G. Mialon, Y. Tian, and et al., "A cookbook of self-supervised learning," Apr 2023. [Online]. Available: https://arxiv.org/abs/2304.12210

[3] C. Thomé, S. Piwell, and O. Utterbäck, "Musical audio similarity with self-supervised convolutional neural networks," *arXiv preprint arXiv:2202.02112*, 2022.

[4] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," 2021.

[5] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006.

[6] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," 2009. [Online]. Available: https://www.jmlr.org/papers/v10/weinberger09a.html

[7] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Byol for audio: Exploring pre-trained general-purpose audio representations," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 137–151, 2023.

[8] H. Al-Tahan and Y. Mohsenzadeh, "Clar: Contrastive learning of auditory representations," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 2530–2538.

[9] A. Bardes, J. Ponce, and Y. LeCun, "Vicreg: Variance-invariance-covariance regularization for self-supervised learning," *arXiv preprint arXiv:2105.04906*, 2021.

[10] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020.

[11] H. Ding, W. Song, C. Zhao, F. Wang, G. Wang, W. Xi, and J. Zhao, "Knowledge-graph augmented music representation for genre classification," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[12] L. Jing, P. Vincent, Y. LeCun, and Y. Tian, "Understanding dimensional collapse in contrastive self-supervised learning," 2022.

[13] M. Won, K. Choi, and X. Serra, "Semi-supervised music tagging transformer," *arXiv preprint arXiv:2111.13457*, 2021.

[14] Y. Gong, A. Rouditchenko, A. H. Liu, D. Harwath, L. Karlinsky, H. Kuehne, and J. R. Glass, "Contrastive audio-visual masked autoencoder," in *The Eleventh International Conference on Learning Representations*, 2022.

[15] J. Lee, J. Park, K. Kim, and J. Nam, "Samplecnn: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, 01 2018.

[16] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "Fma: A dataset for music analysis," Sep 2017. [Online]. Available: https://arxiv.org/abs/1612.01840

[17] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[18] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging," in *International Society for Music Information Retrieval Conference*, 2009.

[19] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere, "The million song dataset." in *International Society for Music Information Retrieval Conference*, 01 2011, pp. 591–596.

[20] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1068–1077.

[21] J. Burgoyne, J. Wild, and I. Fujinaga, "An expert ground truth set for audio chord recognition and music analysis." in *International Society for Music Information Retrieval Conference*, 01 2011, pp. 633–638.

[22] L. Wang, P. Luc, Y. Wu, A. Recasens, L. Smaira, A. Brock, A. Jaegle, J.-B. Alayrac, S. Dieleman, J. Carreira, and A. van den Oord, "Towards learning universal audio representations," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 4593–4597.

[23] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in neural information processing systems*, vol. 33, pp. 12 449–12 460, 2020.

[24] A. Tjandra, D. G. Choudhury, F. Zhang, K. Singh, A. Conneau, A. Baevski, A. Sela, Y. Saraf, and M. Auli, "Improved language identification through cross-lingual self-supervised learning," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 6877–6881.

[25] S. Srivastava, Y. Wang, A. Tjandra, A. Kumar, C. Liu, K. Singh, and Y. Saraf, "Conformer-based self-supervised learning for non-speech audio tasks," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 8862–8866.

[26] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[27] Z. Fan, M. Li, S. Zhou, and B. Xu, "Exploring wav2vec 2.0 on speaker verification and language identification," in *Interspeech*, 2020.

[28] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21 271–21 284, 2020.

[29] V. Chudasama, P. Kar, A. Gudmalwar, N. Shah, P. Wasnik, and N. Onoe, "M2fnet: Multi-modal fusion network for emotion recognition in conversation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2022, pp. 4652–4661.

[30] B. Nguyen, S. Uhlich, and F. Cardinaux, "Improving self-supervised learning for audio representations by feature diversity and decorrelation," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[31] A. Saeed, D. Grangier, and N. Zeghidour, "Contrastive learning of general-purpose audio representations," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3875–3879.

[32] H. Karnan, E. Yang, D. Farkash, G. Warnell, J. Biswas, and P. Stone, "Self-supervised terrain representation learning from unconstrained robot experience," in *ICRA2023 Workshop on Pretraining for Robotics (PT4R)*, 2023. [Online]. Available: https://openreview.net/forum?id=ShKj-4wLWyv

[33] L. Xu, K. D. Mueller, J. Liss, and V. Berisha, "Decorrelating language model embeddings for speech-based prediction of cognitive impairment," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[34] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM International Conference on Multimedia*. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2733373.2806390

[35] K. Fukushima, "Visual feature extraction by a multilayered network of analog threshold elements," *IEEE Transactions on Systems Science and Cybernetics*, vol. 5, no. 4, pp. 322–333, 1969.

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

[37] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, pp. 223–311, 2016.

[38] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[39] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang, and S. Wang, "Image and video compression with neural networks: A review," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, pp. 1683–1698, 2019.

[40] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.

[41] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," Jan

1993. [Online]. Available: https://proceedings.neurips.cc/paper/1993/hash/288cc0ff022877bd3df94bc9360b9c5d-Abstract.html

[42] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6964–6968.

[43] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *ArXiv*, vol. abs/1807.03748, 2018.

[44] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," *ArXiv*, vol. abs/1808.06670, 2018.

[45] Y. You, I. Gitman, and B. Ginsburg, "Large batch training of convolutional networks," 2017.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[47] B. L. Sturm, "The gtzan dataset: Its contents, its faults, their effects on evaluation, and its future use," Jun 2013. [Online]. Available: https://arxiv.org/abs/1306.1461

[48] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," *arXiv e-prints*, Jun. 2016.

[49] D. Yao, Z. Zhao, S. Zhang, J. Zhu, Y. Zhu, R. Zhang, and X. He, "Contrastive learning with positive-negative frame mask for music representation," in *Proceedings of the ACM Web Conference 2022*. ACM, apr 2022. [Online]. Available: https://doi.org/10.1145%2F3485447.3512011

[50] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," 2020.

[51] Y. Tian, X. Chen, and S. Ganguli, "Understanding self-supervised learning dynamics without contrastive pairs," 2021.

[52] J. Pons and X. Serra, "musicnn: Pre-trained convolutional neural networks for music audio tagging," *arXiv preprint arXiv:1909.06654*, 2019.

[53] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[54] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

[56] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[57] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," *Advances in neural information processing systems*, vol. 33, pp. 22 243–22 255, 2020.

[58] C. Niu and G. Wang, "Self-supervised representation learning with multi-segmental informational coding (music)," 2022.

[59] A. Bardes, J. Ponce, and Y. LeCun, "Vicregl: Self-supervised learning of local visual features," 2022.

[60] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 310–12 320.

A

Extended Results

This appendix contains supplementary results to the report. For additional results, see Additional Results B.

Table A.1: Comparison of the ROC-AUC and PR-AUC scores for the CLMR and VICReg at different training epochs. The scores achieved using a single-layer final head are shown, with the scores achieved using a two-layer final head are shown in parentheses. The table tracks the performance over extended training, providing insights into the convergence behavior of the two models.

	CLMR		VICReg	
Epochs	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
300	87.52 (88.12)	33.40 (33.83)	87.08 (88.19)	32.24 (33.55)
1000	86.83 (87.47)	31.88 (32.55)	87.02 (88.35)	32.79 (33.95)
3000	86.69 (87.41)	31.46 (32.26)	87.73 (88.65)	33.88 (34.80)
5000	86.75 (87.39)	31.70 (32.43)	87.96 (88.96)	34.40 (35.49)
8000	86.22 (87.48)	31.33 (32.73)	87.93 (89.14)	34.64 (35.47)

Table A.2: Performance of VICReg and CLMR architectures as a function of representation layer size. The table presents ROC-AUC and PR-AUC scores for representation layer sizes of 32, 64, 128, 256 and 512. For each architecture and representation size, the scores achieved using a single-layer final head are shown, with the scores achieved using a two-layer final head shown in parentheses. All backbones are trained for $1,000$ epochs.

	VICReg		CLMR	
Representation	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
32	77.19 (84.40)	21.46 (27.14)	83.01 (84.42)	26.47 (26.73)
64	81.08 (86.20)	25.03 (29.87)	85.60 (86.32)	30.11 (30.75)
128	84.74 (87.01)	28.64 (31.20)	86.26 (87.24)	30.77 (31.94)
256	86.24 (87.74)	30.87 (33.24)	86.96 (87.71)	32.03 (32.81)
512	87.21 (88.37)	32.56 (34.03)	87.07 (87.82)	32.46 (33.22)

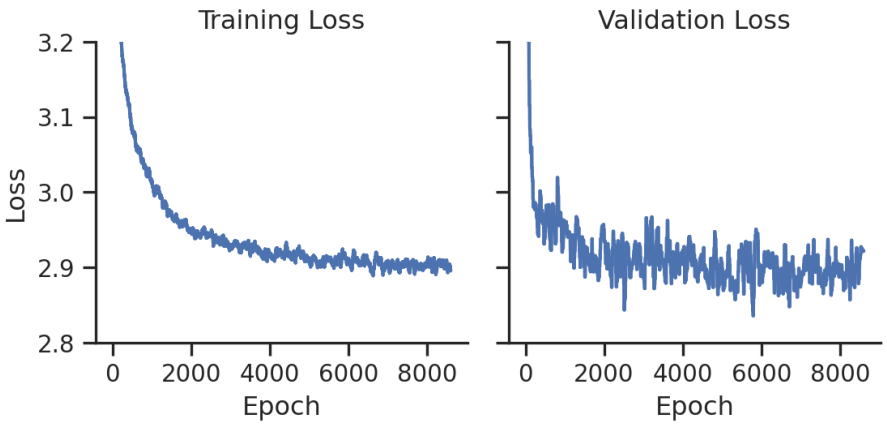

Figure A.1: Training and validation loss for CLMR measured over $8,000$ epochs. The training dataset is FMA, and the validation dataset is MTAT. The losses are smoothed over 50 epochs. Note that the validation loss is evaluated every 20 epochs; therefore, each point in the validation loss is only smoothed over 3 values.

B

Additional Results

In our primary investigation, we focused on the performance of VICReg compared to CLMR models. However, during this process, we also conducted an array of additional experiments that, while not central to our research question, surfaced interesting patterns and insights. We have collated these findings in this 'Additional Results' section.

First, we explored how the performance of the models is influenced by batch size. While batch size is often a practical constraint based on available computational resources, it also impacts model training dynamics and, ultimately, the quality of learned representations. Particularly for VICReg, its Variance term is calculated along the batch dimension; therefore, larger batch sizes encompass larger sample sizes for computing variance.

Secondly, we explored the impact of different projectors on the models' performance. The choice of the projector, which maps the raw representation to a lower-or-higher-dimensional space, can crucially affect the efficacy of the learning process. More specifically, we interchange the projectors of VICReg [9] and CLMR [4]. Our findings delineate the role of different projectors in the context of these methods.

Finally, we conducted a grid search over batch size and representation size on FMA and evaluated using PR-AUC on MTAT.

B.1 VICReg batch size

We perform a search over batch size for VICReg, keeping a constant representation size of 512. The pre-training was carried out over $1,000$ epochs on the FMA dataset. Following this, we trained one or two fully connected layers atop the frozen representations of SampleCNN on the MTAT dataset. See Figure B.1 for a graphical representation of performance as a function of batch size and Table B.1 for corresponding data points. Additionally, the progression of VICReg loss terms over epochs is illustrated in Figure B.2.

The data depicted in Figure B.1 and Table B.1 demonstrates that VICReg's optimal performance, using SampleCNN as a backbone with a representation size of 512, is achieved with a batch size of 768. The only exception is PR-AUC when using a two-layer final head. Our optimal batch size is of similar magnitude to its representation

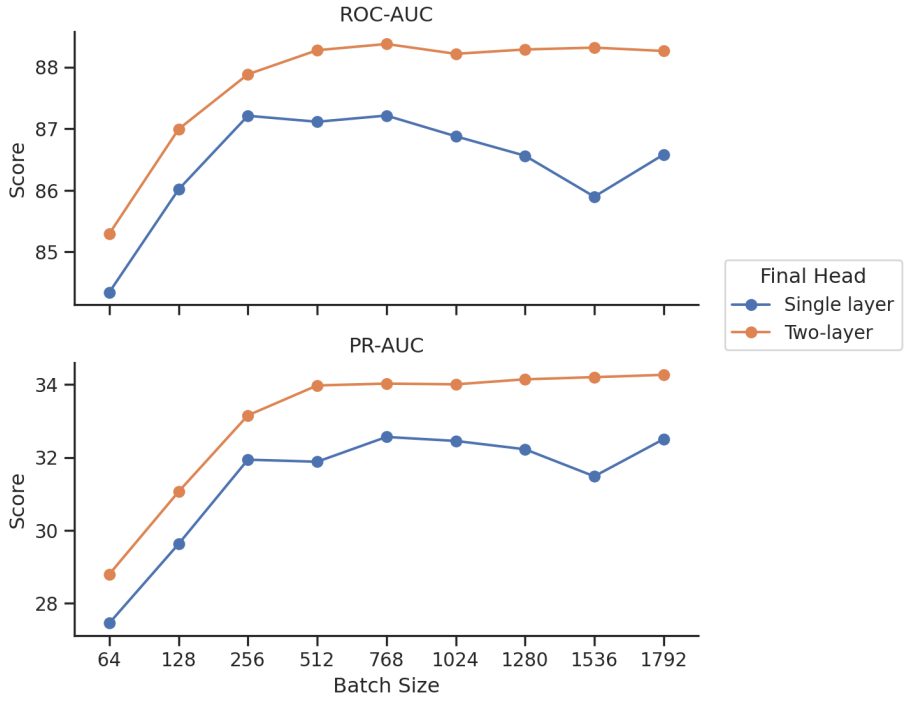

Figure B.1: VICReg performance on MTAT with different batch sizes using a representation layer of 512. Performance is measured in ROC-AUC and PR-AUC.

size; this aligns with the finding of Bardes et al. [9]. For further analysis regarding batch size and representation dimension, see Section B.3.

In Figure B.2, we illustrate how the regularization terms of VICReg evolve across epochs. With results from Figure B.1 and Table B.1, it is evident even though that a batch size of 768 yielded the best result; it is not the lowest in any of the terms. This suggests that the optimal solution is a balance of all terms. Remember that the Variance is calculated along the batch dimension; hence, larger batch sizes also increase this term. Therefore this term might not be directly comparable.

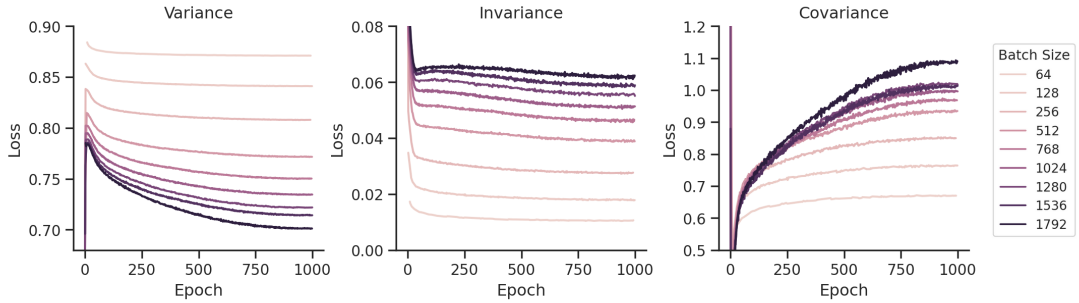

Figure B.2: VICReg loss terms over 1,000 epochs using different batch sizes.

Batch Size	VICReg	
	ROC-AUC	PR-AUC
64	84.35 (85.30)	27.46 (28.80)
128	86.01 (86.99)	29.63 (31.06)
256	87.21 (87.88)	31.94 (33.15)
512	87.11 (88.27)	31.88 (33.97)
768	**87.21** (**88.37**)	**32.56** (34.03)
1024	86.88 (88.21)	32.45 (34.01)
1280	86.56 (88.29)	32.22 (34.14)
1536	85.90 (88.32)	31.48 (34.20)
1792	86.58 (88.26)	32.50 (**34.27**)

Table B.1: VICReg performance on MTAT with different batch sizes using a representation layer of 512. Performance is measured in ROC-AUC and PR-AUC. Scores in parentheses are obtained using a two-layer MLP. Bold numbers represent the highest score achieved in each category.

B.2 Projectors

In this experiment, we compare the projector employed in VICReg[9] and CLMR[4]. CLMR's projector features a network of 512-512-128 neurons, whereas VICReg's employs a more complex network of 512-8196-8196-8196 neurons. We train the VICReg and CLMR backbones over 1,000 epochs on the FMA dataset. Following this, we train a single/two fully connected layer/s on top of the frozen representations of SampleCNN on the MTAT dataset. The results are in Table B.2. Here, numbers in parentheses represent the performance using the two-layer fully connected network, and bold numbers indicate the best result within their respective category.

Regarding the ROC-AUC category, the CLMR backbone with the CLMR projector achieved the highest score when a single layer was used: 87.07. However, the VICReg backbone with the VICReg projector surpassed this score when a two-layer fully connected network was applied, achieving a score of 88.22. In the PR-AUC category, The VICReg backbone with the VICReg projector outperformed the other combinations in single- and two-layer configurations, scoring 32.82 and 33.71, respectively.

The results suggest that the representations learned by the CLMR backbone are more linearly separable than those learned by the VICReg backbone. This is evidenced by the fact that the CLMR backbone with the CLMR projector achieved the highest ROC-AUC score when only a single layer was used. However, when a second layer was added, the VICReg backbone with the VICReg projector achieved a higher ROC-AUC score, indicating that the additional layer may be helping to untangle more complex, non-linear relationships in the VICReg representations.

It is interesting that CMLR works better with smaller projections while VICReg does not. VICReg performs better with a wider and deeper MLP as projection [9],

same as in Barlow Twins [60]. Contradictory to our results, this should also hold true for CLMR since it has been shown that using NT-Xent with a deeper projection head improves performance [57]. We do not know why this is the case, but further studies regarding this could prove interesting.

Table B.2: Comparison between using CLMR and VICReg Projectors on each respective backbone. Scores in parenthesis are trained using two fully connected layers. Bold numbers represent the highest score achieved in each category. All backbones are pre-trained on FMA, and linear heads are trained on MTAT.

	CLMR Projector		VICReg Projector	
	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
CLMR	**87.07** (87.82)	32.46 (33.22)	86.81 (87.80)	31.94 (32.92)
VICReg	85.70	30.21	86.97 (**88.22**)	**32.82** (**33.71**)

B.3 Batch size vs. Representation dimension

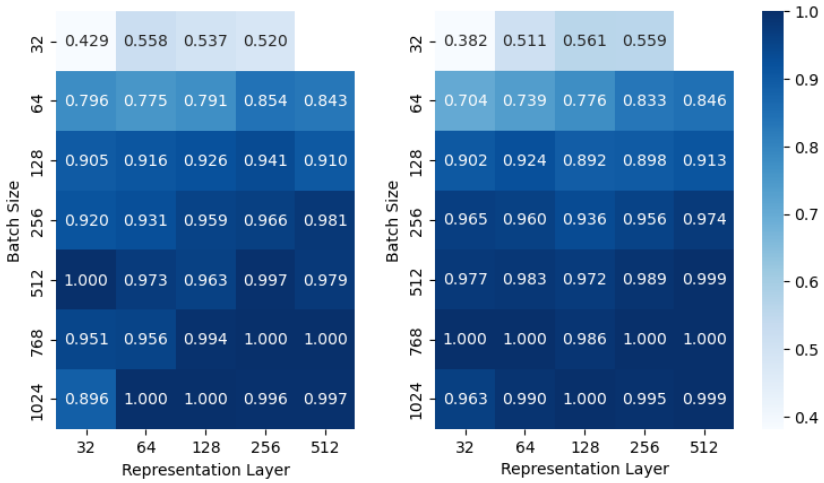

Figure B.3: Heatmap of normalized PR-AUC scores on MTAT using VICReg pre-trained on FMA. The scores are normalized column-wise, that is, by the representation layer. The left figure is the result using a one-layer linear head, right using a two-layer head. Each backbone is pre-trained for 1,000 epochs.

Our findings challenge the prevalent perspective that the optimal batch size for training is intricately tied to the dimension of the representation layer. Such an assumption has been observed in various studies, for instance, the study by VICReg, which achieved superior performance utilizing a batch size of 2,048 while operating a Resnet-50 with an output dimension of 2,048 [9]. Likewise, in their usage of ConvNeXt-S (representation size of 768) and ConvNeXt-B (representation size of 1,024), they determined that an optimal batch size of 512 led to the best results [59].

This again alluded to a relationship between optimal batch size and representation dimension.
In contrast, our investigation with VICReg suggests that the optimal batch size does not exhibit such a dependency on the representation dimension. When using the same model but varying representation size, the optimal batch size is about 768, even with a representation size of 32, see Figure B.3. This suggests that rather than representation size, the architecture of the network may play a more consequential role in determining the optimal batch size for training.
In light of this, the conventional "one-size-fits-all" approach to selecting batch size may be an oversimplification, potentially limiting the performance and effectiveness of various models. This finding advocates for a more nuanced approach in selecting the batch size, where the specific architecture of the neural network is given due consideration. However, further exploration with different models would need to be experimented to confirm this.

Department of Architecture and Civil Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY