University of Amsterdam

MSc Artificial Intelligence
Master Thesis

Contrastive Learning of Musical Representations

---

by
Janne Spijkervet
10879609

Friday 15th January, 2021

Number of Credits
48

Supervisor:
Dr. J.A. Burgoyne

Assessor:
Dr. W. Aziz

Faculty of Science

University of Amsterdam

Abstract

Learning and designing representations lie at the heart of many successful machine learning tasks. Supervised approaches have seen widespread adoption within music information retrieval for learning such representations, but unsupervised representation learning remains challenging. In this thesis, we combine the recent insights of self-supervised learning techniques and advances in representation learning for audio in the time domain, and contribute a chain of data augmentations and their effectiveness in an ablation study, together to form a simple framework for self-supervised learning of raw, musical audio: *CLMR*. This approach requires no manual labeling, no fine-tuning and no pre-processing of audio data to learn useful representations. We evaluate the self-supervised learned representations in the downstream task of music classification on the MagnaTagATune and Million Song datasets. A *linear* classifier fine-tuned on representations from a frozen, pre-trained CLMR model achieves a score of 35.4% PR-AUC on the MagnaTagATune dataset, superseding fully supervised models that currently achieve a score of 34.9%. Moreover, we show representations learned by CLMR from large, unlabeled corpora are transferable to smaller, labeled musical corpora, indicating that they capture important musical knowledge. Lastly, we show that when CLMR is fine-tuned on only 1% of the labels in the dataset, we still achieve 33.1% PR-AUC despite using 100× fewer labels. To foster reusability and future research on self-supervised learning in MIR, we publicly release the pre-trained models and the source code of all experiments of this thesis.[1]

[1] https://github.com/spijkervet/CLMR

Acknowledgements

This thesis was written in an extraordinary year, during which I received a great deal of support.

I would first like to thank my supervisor, dr. John Ashley Burgoyne, for sharing his expertise and aptitude for performing multidisciplinary research in music. Your invaluable insights and feedback brought this work to a higher level - not to mention your firm commitment and support during the last few days before the ISMIR paper deadline. I would also like to thank him for introducing me to many of the fantastic people in MIR research during ISMIR 2019 in Delft, which was both a stepping-stone for this research and my research career. Your teaching, guidance, support and the "AI Song Contest adventure" are experiences I will cherish.

I would also like to thank my mentor, dr. Jordan B.L. Smith for his support and guidance throughout 2020, and his help to shape the accompanying research paper's final review version for ISMIR.

I would also like to acknowledge all my colleagues from the MSc Artificial Intelligence of the University of Amsterdam. I would like to thank them for the many great discussions, fun activities and study sessions: without them, this incredible subject would have been so much harder to understand.

In addition, I would like to thank my parents for their love and support. Thank you for always being there for me.

Finally, I could not have completed this work without my friends, who provided love, valuable discussions and joyful distractions during a year unlike any other.

January, 2021

Contents

1 Introduction

The field of music information retrieval (MIR) has seen many successes since the emergence of deep learning. Supervised, end-to-end learning methods have been widely used in tasks like chord recognition (Chen and Su, 2019; Korzeniowski and Widmer, 2016), key detection (Korzeniowski and Widmer, 2017), beat tracking (Böck et al., 2016), music audio tagging (Pons et al., 2017) and music recommendation (van den Oord et al., 2013). These methods use labeled corpora, which are hard (Koops et al., 2019), expensive and time-consuming to create, while raw unlabeled musical data is available in vast amounts. Despite the importance of unsupervised learning in MIR for raw, high-dimensional signals of audio, it has yet to see breakthroughs similar to supervised learning. It has enjoyed successes with methods like PCA, PMSC's and spherical k-means that rely on a transformation pipeline (Dieleman and Schrauwen, 2013; Hamel et al., 2011), but learning effective representations of raw audio remains elusive.

Self-supervised representation learning, a form of unsupervised learning, is a relatively new, upcoming learning paradigm (Chen et al., 2020a; Dosovitskiy et al., 2015; Hjelm et al., 2019; Oord et al., 2019). The general goal of representation learning is to train a function g that maps input data $x \in \mathbb{R}^d$ to some representation of lower dimensionality, while preserving as much useful information as possible. In the absence of ground truth, there can be no ordinary loss function for training g; self-supervised learning trains by way of a proxy loss function instead, obtained by withholding or augmenting parts of the input data. One way to preserve the amount of useful information during self-supervised learning is to define the proxy loss function with respect to a relatively simple 'pretext' task, with the idea that a representation that is good for the pretext task will also be useful for other tasks. Many approaches simply rely on heuristics to design pretext tasks (Doersch et al., 2015a; Zhang et al., 2016a), e.g., by defining pitch transformation as a pretext task (Gfeller et al., 2020). Alternatively, *contrastive representation learning* formulates the proxy loss directly on the learned representations and relies on comparing and contrasting multiple, differing versions or neighbouring patches of any one example. The rationale behind this contrastive strategy is *predictive coding*, a theory that the human brain encodes causal structures and predicts future events at different levels of abstraction (Friston and Kiebel, 2009).

In this thesis, we combine the insights of contrastive learning techniques and recent advances in representation learning for audio in the time domain, and contribute a pipeline of data augmentations on raw audio, to form a simple framework for self-supervised, con-

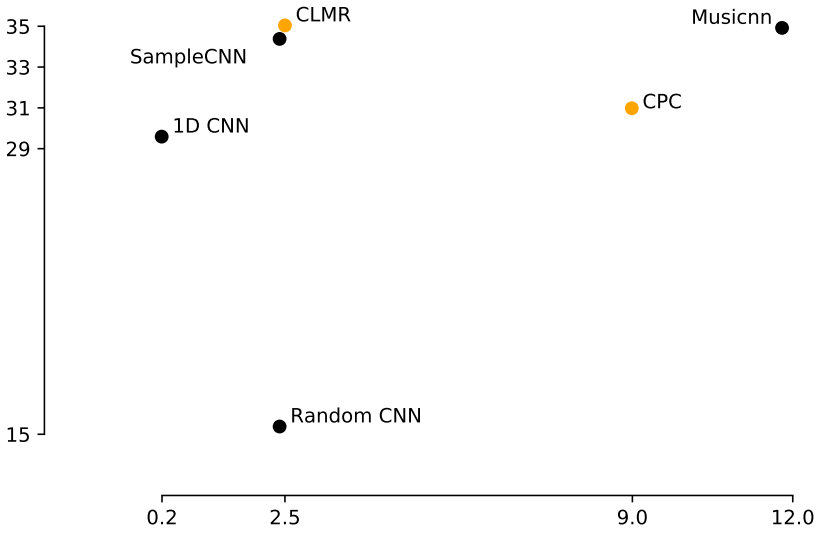


Figure 1.1: Performance and model complexity comparison of supervised models (grey) and self-supervised models (ours) in music classification of raw audio waveforms on the MagnaTagATune dataset to evaluate musical representations. Supervised models were trained end-to-end, while CLMR and CPC are pre-trained without ground truth: their scores are obtained by training a linear classifier on their learned representations but nonetheless perform better than the supervised models.

trastive learning of representations of raw audio waveforms. To compare the effectiveness of this simple framework compared to a more complex self-supervised learning objective, we also evaluate representations learned by contrastive predictive coding (Oord et al., 2019). The models are evaluated on the downstream music tagging task, enabling us to evaluate their versatility: music tags describe many characteristics of music, e.g., genre, instrumentation and dynamics. Our key contributions are summarized as follows.

- CLMR achieves strong performance on the music classification task, despite self-supervised pre-training and fine-tuning on the downstream task using a linear classifier (see Figure 1.1).
- CLMR learns useful, compact representations from raw signals of musical audio.
- CLMR enables efficient classification: compared to fully supervised models, when fine-tuning a linear classifier with the self-supervised learned representations on the task of music classification, we achieve comparable performance using as few as 1% of the labeled data.
- The learned representations are transferable across different musical corpora.
- CLMR can learn from *any* dataset of raw audio, requiring neither transformations nor fine-tuning on the input data; nor do the models require manually annotated labels for pre-training.
- We provide a thorough ablation study on the effectiveness of the data augmentations of raw audio on the downstream performance of music tagging.

1.1 Outline

In the following chapter, a comprehensive background of the field of self-supervised learning is be presented, as well as neural network architectures commonly used in the raw audio domain. Subsequently, the main downstream task and application of this thesis will be elaborated upon, along with its evaluation metrics.

In chapter 3, the method of this thesis is presented, outlining the details and intuition of the architecture of the CLMR model.

In chapter 4, the details of the implementation of the framework is discussed.

In chapter 5, the datasets are outlined, both for the main and transfer learning experiments.

In chapter 6, the results of the linear classification, efficient classification and transfer learning tasks, along with the study on the effect of different data augmentations, parameters and mini-batch sizes on the performance of the downstream task, are presented.

In chapter 7, we give a thorough qualitative analysis of the learned representations of the CLMR model. Rather than looking at hard numbers, we perform a listening experiment to gain a deeper understanding of the representations that is has learned.

Lastly, the conclusions of this thesis are presented in chapter 8.

2 Background

In this chapter, we recapitulate the foundations which are used to build on in this work. In Section 2.1, we discuss common approaches to representation learning. We first discuss the intuition behind an upcoming learning paradigm, self-supervised learning, in Section 2.2 and highlight the literature and challenges on self-supervision on audio data in Section 2.3. After providing a general intuition behind this learning method, we dive into self-supervised contrastive learning in section 2.4, in which a gentle theoretical foundation is provided and two self-supervised contrastive learning frameworks are discussed. In Section 2.5, we discuss convolutional neural networks for raw audio signals. Finally, we discuss the task of music auto tagging and their evaluation metrics in Section 2.6.

2.1 Representation Learning

The goal of representation learning is to identify features that make prediction tasks easier and more robust to the complex variations of natural data (Bengio et al., 2013). Supervised techniques for representation learning have now been successfully applied to a variety of tasks in the audio domain, e.g. rhythm detection, musical key recognition, chord recognition, music auto tagging, speaker identity recognition and phoneme recognition (Böck et al., 2016; Chen and Su, 2019; Korzeniowski and Widmer, 2016,1; Pons et al., 2017; van den Oord et al., 2013). We first give a brief overview of three main strategies of learning:

- **Supervised learning** uses (often) human-annotated labels as a guidance to learning an objective. This learning paradigm is (often) limited to the amount of manually annotated data that is available for training.
- **Unsupervised learning** is the set of algorithms that do not exploit pre-annotated labels.
- **Reinforcement learning** uses agents that maximise their reward in an environment by performing sequences of actions to learn an objective. Learning is unguided in the sense that suboptimal actions are not explicitly corrected by a supervisory signal.

We also highlight the difference between between two learning methods that fall under the unsupervised learning paradigm to avoid confusion:

- Semi-supervised learning is a learning paradigm that manifests itself between unsupervised and supervised learning. It uses few supervisory signals to guide training using many unlabeled data points.

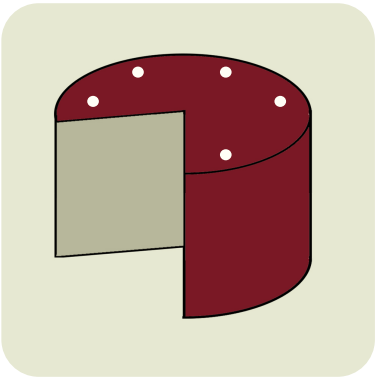


Figure 2.1: Yann LeCun, a strong advocate of unsupervised learning, famously introduced the 'cake analogy' at NIPS 2016: *"If intelligence is a cake, the bulk of the cake is unsupervised learning, the icing on the cake is supervised learning, and the cherry on the cake is reinforcement learning."*

- Self-supervised learning is a form of unsupervised learning that formulates the learning objective in such a way, that it retrieves a supervisory signal from (transformations of) the data itself.

The distinguishment between learning methods in the unsupervised learning domain can be daunting at first, e.g., generative modeling and likelihood-based models are considered unsupervised learning methods that typically find useful representations of the data by attempting to reconstruct the observations on the basis of their learned representations (Goodfellow et al., 2014; Radford et al., 2016). Broadly speaking, these approaches can also be considered as self-supervised representation learning: the objective is formulated in such a way that it gets supervision from the data itself. The difference between generative approaches and self-supervised learning we like to distinguish, is that self-supervised learning aims to identify the explanatory factors of the data using an objective that is formulated with respect to the representations directly, and its goal is not to generate a faithful reproduction of the data but rather to learn useful features for (multiple) downstream tasks.

In this thesis, our focus will be on the *self-supervised* learning paradigm. We will first give an overview of its usage in different domains, as to sketch a clear image of its workings and the unique challenges in each domain, while work on self-supervised learning in audio is relatively limited.

2.2 *Self-supervised Learning*

This idea of self-supervised learning has seen widespread adoption in language modeling. Its most common task is to predict the next word, given a past sequence of words, but more auxiliary (pretext) tasks can be added to improve the language model. For example, BERT (Devlin et al., 2019) adds two auxiliary tasks that both rely on self-generated labels to improve the bi-directional prediction of word tokens and sentence-level understanding: 1) a cloze test (Taylor, 1953), in which part of the tokens in each sequence is randomly masked and the model is asked to predict the missing tokens, and 2) optimising a binary classifier on predicting whether one sequence follows another sequence. The first pretext task encourages the model to better capture the syntactic and semantic meaning of the context around a word, and the second task improves the understanding of relationships between sentences. Building these tasks requires no manual labeling, and can therefore be scaled up to arbitrary size while there is plenty of free text available to use as training data.

2.2.1 *Formulation of pretext tasks*

In the image domain, self-supervised learning manifests itself in a similar way: one or multiple pretext tasks are formulated on a set of unlabelled images and, subsequently, the pre-trained encoder or its intermediate layers are used to fine-tune on a downstream task like image classification. We first resort to the image domain that has enjoyed more attention than the field of audio research to give a more clear intuition of the types of pretext tasks that were devised to learn useful representations for downstream tasks. We give a short description of five different approaches.

Exemplar-CNN (Dosovitskiy et al., 2014) creates a surrogate dataset by randomly applying a sequence of transformations to 'exemplary' image patches that contain large gradients, e.g., image patches that contain edges, strong textures, i.e., objects or parts of objects of interest. Its pretext task is to classify the corresponding class of each transformed image.

An even more simple pretext task is formulated in RotNet (Gidaris et al., 2018), that proposes to use a random 2d rotation transformation as a supervisory signal to learn semantic features of an image. The image is randomly rotated and given a class label: no rotation, 90°, 180° or 270°, making the pretext task a 4-class classification problem. Arguably, this forces the model to learn relationships in the semantic space of objects, i.e., to recognize the same image under different rotations, it has to learn more high-level, structural parts of the image, e.g., the relative position of a nose with respect to the eyes. RotNet drastically reduced the gap between unsupervised and supervised feature learning in the image domain using a simple pretext task.

Another common transformation is that of colorization (Zhang et al., 2016b), in which they extracted the lighting channel L from a colored image and subsequently asked the model to predict the corresponding a and b color channels in CIE *Lab* colorspace.

A pretext task can also be formulated as a relationship between two random patches of a single image. In (Doersch et al., 2015b), they exploit the spatial context of an image as a supervisory signal. Again given a large set of unlabeled images, random pairs of patches are extracted from each image and the network is asked to predict the position of the second patch relative to the first patch[1]. A 3×3 grid is constructed and, given the first patch is located in the center, the model is asked to predict the location of a patch located in any of the remaining 8 positions, turning the pretext task into an 8-class classification problem.[2]

To conclude self-supervised learning in the image domain, (Noroozi and Favaro, 2016) converted aforementioned pretext task into a full 3×3-grid 'jigsaw puzzle', asking the model to reconstruct a sampled patch of an image after randomly shuffling all 9 sub-patches.

[1] In contrast to Exemplar-CNN, sampling is done without regard to the content of the image.

[2] Interestingly, this approach quickly found a trivial solution to the problem of identifying the relative position between a pair of images: chromatic aberration. This phenomenon arises when a lens fails to focus light at different wavelengths (Brewster and Bache, 1835). Convolutional neural networks are able to localise such patches relative to the lens, which makes the objective of identifying the relative position between two patches very easy to solve. While detailing more potential trivial solutions in the image domain is beyond the scope of this thesis, it is important to note that care must be taken for the model's ability to find trivial solutions to the problem when designing a pretext task. In this case, the trivial solution was mitigated by shifting the green and magenta color channels to gray.

From these series of approaches to self-supervised learning, we like to distinguish two categories of pretext tasks throughout this thesis: those that involve distortions to learn **spectral relationships** and those that use patches to learn **spatial relationships** in data.

2.3 *Self-supervision on Audio*

Self-supervised learning on audio brings unique challenges compared to the image domain. Audio signals are high-dimensional, have a variable-length, and entail a hierarchical structure that is hard to infer without a supervisory signal. It is also highly variable, given different recording conditions, voice types, instrumentation, phonemes, syllables, etc. Work on self-supervised learning in audio was very limited at the beginning of this thesis. While it is still very limited in the music information retrieval field, several papers were published in the speech domain.

PASE proposed a multi-task self-supervised learning approach, in which several workers each solved a self-supervised task for one neural encoder (Pascual et al., 2019). The learned representations were proven useful for speaker identity, phoneme and emotional cue recognition.

During this thesis, PASE+ was published and improved on the latter method by adding random transformations to raw audio signals for more robust representations under noisy and reverberant recording environments (Ravanelli et al., 2020). It outperforms both PASE and encoders trained using common audio features, like MFCC's and filter banks. These series of data augmentations for audio will be further elaborated in Section 2.3.2.

The workers in the PASE papers are small feed-forward neural networks and both solve self-supervised tasks. Common speech features are extracted from the audio, and are used as supervisory signals for the workers. These include regression workers that estimate log-power spectra, MFCCs, prosody features, filter banks and their derivatives. Other workers are simple binary classifiers trained to maximize the mutual information between representations of positive and negative samples. The encoder and workers are jointly optimised using a loss function that is formulated as the mean of workers' cost.

Interestingly, the self-supervised learned features are also transferable: when trained on the LibriSpeech dataset, it achieves 74.1% WER on the highly challenging CHiME-5 task (Barker et al., 2018).

Contrastive predictive coding (CPC) was introduced as a universal approach to self-supervised learning, and has been successful for speaker and phoneme classification using raw audio, among other tasks in different domains (Oord et al., 2019). It will be further detailed in Section 2.4.2.

In music information retrieval specifically, recent advances have been

made in self-supervised pitch estimation (Gfeller et al., 2020), closely matching supervised, state-of-the-art baselines despite being trained without ground truth labels. Given a segment of raw audio, it scales the pitch of the signal, converts it to the time-frequency domain using a CQT transform as input data for a ConvNet encoder, and uses the scaling factor as a supervisory signal. To the best of our knowledge, SPICE (Gfeller et al., 2020) is the only (peer-reviewed) paper on self-supervised learning on audio in music information retrieval at the publication date of this thesis.

We are the first to perform self-supervised learning on raw audio waveforms of musical audio, without a transformation pipeline to the time-frequency domain, and evaluate the learned representations in a musical, downstream task.

2.3.1 *Ideal Representations*

The aforedescribed pretext tasks are designed in a way that they allow a model to learn representations that are not limited to solving the pretext task, but are also helpful in solving the downstream task when fine-tuning a classifier using the pre-trained intermediate layers as feature extractors. Ideal feature representations should be invariant to local translations and noisy variations of the input signal while remaing sensitive to higher-level semantic information.

Put differently, the main challenge is to learn representations that effectively encode *slow features* (Wiskott and Sejnowski, 2002), i.e., the shared information between parts of a high-dimensional signal. Conversely, a good representation should disregard noisy, more local features. The idea of slow features is quite intuitive for music. We know that an audio fragment of a few seconds will share information with neighbouring fragments, e.g., the instrument(s) playing, the harmonic set of pitches or the identity of a vocalist. But the further into the future a model is forced to predict these features, the less of this kind of shared information is available, thereby requiring the model to infer higher-level structure. Slow audio features span a longer temporal range (e.g., harmonic transitions or melodic contour), or a larger spectral range (e.g., the frequency range, loudness) and are more interesting for use in downstream MIR tasks.

2.3.2 *Audio Augmentations*

Earlier we distinguished two categories of pretext tasks: those that learn spectral relationships and spatial relationships. We can extend this intuition to data augmentations, as is done in Exemplar-CNN (Dosovitskiy et al., 2014) and PASE+ (Ravanelli et al., 2020) as to create surrogate samples or learn more robust representations respectively.

As described in the previous section, designing pretext tasks and augmentations in the audio domain brings unique challenges. We reckon one could resort to the time-frequency domain and use spec-

tograms or CQT-transforms and treat them as visual input data, but one could argue that aforedescribed augmentations and pretext tasks have little to do with the spectral and spatial dynamics of an audio signal, e.g., randomly flipping a spectogram or applying color jitter to a CQT-transform has hardly anything to do with the original audio signal. We therefore describe several 'spectral', i.e., acoustic augmentations that were introduced in the self-supervised speech representation learning literature (Ravanelli et al., 2020) in Table 2.1.

Augmentations of musical data are motivated by the observation that learning algorithms may generalise better and learn more robust representations when trained on samples that are perturbed (McFee et al., 2015). The augmentations introduced in the MUDA framework for musical data augmentations is further described in Table 2.2. In Chapter 3, the audio augmentations used in the experiments of this thesis will be discussed.

Augmentation	Details
Reverberation	Convolution with a large set of impulse responses derived with the image method.
Additive Noise	Non-stationary noises
Frequency Masking	Convolution with band-stop filters, randomly dropping a spectrum band.
Temporal Mask	Replace a random sequence of samples with zeros.
Clipping	Add a random amount of saturation to simulate audio clipping conditions.
Overlapping	Overlap a random sample of audio to the current audio signal.

Table 2.1: Audio augmentations used in the speech domain to learn more robust representations using self-supervised learning methods.

Augmentation	Details
Pitch Shift	Shift the frequency of the signal by $n \in \{-1, 0, +1\}$ semitones.
Time Stretch	Stretch the audio signal by a factor of $r \in \{-2^{\frac{1}{2}}, 1, 2^{\frac{1}{2}}\}$
Background noise	Noise under three pre-recorded conditions is linearly mixed with the input signal y, with α being a random weight: $y' \leftarrow (1-\alpha) \cdot y + \alpha \cdot y_{\text{noise}}$
Dynamic range compression	A common audio signal operation that both amplifies quiet and reduces loud sounds, effectively reducing the signal's dynamic range.

Table 2.2: Musical audio augmentations from (McFee et al., 2015)

2.4 *Contrastive Learning*

Now that the intuition behind self-supervised learning is more clear, we continue to lay out the form of loss functions generally used in self-supervised *contrastive* learning methods: the InfoNCE objective as introduced by (Oord et al., 2019) and its variations. We then provide the details of two frameworks for contrastive learning used in this thesis.

2.4.1 *Contrastive Loss*

The initial form of the contrastive loss function, as introduced by (Hadsell et al., 2006), runs over pairs instead of over individual samples. It was reformulated in (Gutmann and Hyvärinen, 2010) before it was adopted in the self-supervised learning domain by (Oord et al., 2019). While loss functions closely related to contrastive learning were introduced like margin and triplet loss (Chechik et al., 2009; Liu et al., 2016), their differences lie in the sampling strategy of positive and negative samples. In supervised metric learning, the positive samples are chosen from the same class, while negative samples are chosen from different classes utilising hard-negative mining (Sermanet et al., 2017). In a triplet loss function, an input sample is compared to one positive and one negative sample. The choice of positive and negative samples in these losses is guided by the samples' corresponding labels in a supervised setting. Contrastive losses rely on one positive pair, which can either be picked from neighbouring patches of the anchor sample,[3] or an augmented version of the same data point. Different from the other loss functions, contrastive loss functions require many negative samples that are sampled from different data points. Inherently, it is assumed this reduces the probability of a false negative. The initial InfoNCE loss introduced by (Oord et al., 2019) and shown in 2.1 uses a mutual information critic (function f in eq. 2.1) as a similarity metric between positive and negative samples. Variations of NCE-type losses appeared in later work (Hjelm et al., 2019).

[3] The anchor sample and the positive sample together form the positive pair.

$$\mathcal{L}_{\mathrm{N}} = -\mathop{\mathbb{E}}_{X}\left[\log \frac{f_k\left(x_{t+k}, c_t\right)}{\sum_{x_j \in X} f_k\left(x_j, c_t\right)}\right] \tag{2.1}$$

We consider three types of samples[4] in the contrastive loss: the anchor sample, z_i, the positive sample, z_j, and the negative samples $\{z_0 \dots z_{2N-2}\}$. The function f is a scoring function, that could measure, e.g., the mutual information, dot product, cosine similarity or euclidean distance between two samples. The temperature scaling parameter τ controls the penalty of hard-negative samples.

[4] The use of the word 'sample' with regard to this equation may be misleading: the z terms are often embeddings of the input that passed through a parameterised function, i.e., the similarity function is formulated in latent space.

Intuitively, the loss decreases when the scoring function for the positive pair in the nominator increases, and when the similarity between the anchor sample and the negative samples decreases.

2.4.2 *Contrastive Predictive Coding*

Contrastive predictive coding learns useful representations by maximising mutual information among temporally neighbouring patches of data. For audio, it learns to predict representations of future observations from past observations, i.e., it predicts representations of segments of audio in the future, given representations from past sequences. A sequential input signal x_t is mapped by a non-linear encoder $g_{\mathrm{enc}}(\cdot)$ to a sequence of latent representations $h_t = g_{\mathrm{enc}}(x_t)$. Subsequently, the autoregressive model $g_{\mathrm{ar}}(\cdot)$ summarizes all encod-

ings $h_{\leq t}$ in the latent space and maps them to a context latent representation $c_t = g_{\text{enc}}(h_{\leq t})$. A visual overview of CPC is shown in Figure 2.2.

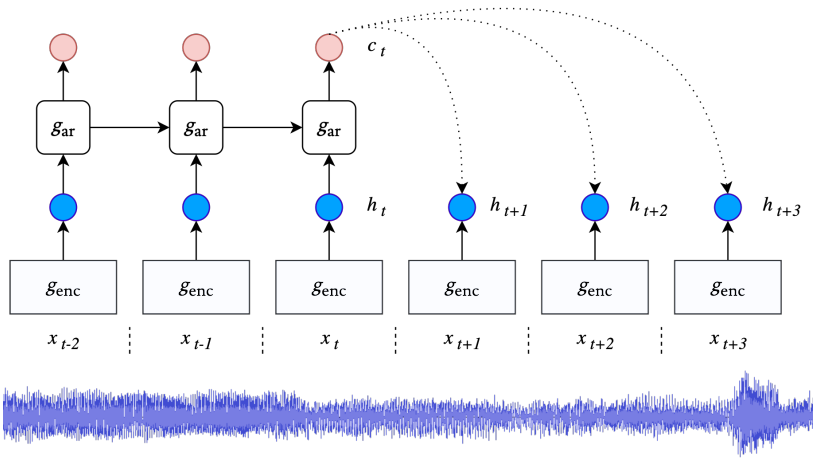


Figure 2.2: Contrastive Predictive Coding jointly optimises two neural networks: a non-linear encoder g_{enc} and an autoregressor g_{ar}, by contrasting the embeddings of temporally neighbouring patches of data using the InfoNCE loss.

The vectors h_t and c_t are encoded so as to preserve maximal mutual information and to identify the shared latent variables of the original signals. The neural networks $g_{\text{enc}}(\cdot)$ and $g_{\text{ar}}(\cdot)$ jointly optimise the InfoNCE loss, a contrastive loss that follows the principles of noise-contrastive estimation (Gutmann and Hyvärinen, 2010). Their principles are widely used in the design of self-supervised loss functions (Chen et al., 2020a; Oord et al., 2019; Sohn et al., 2020).

Given N random samples from the set of encodings $X = \{h_{t+k}, h_{j_1}, h_{j_2} \dots h_N\}$, k being the number of timesteps the encoding occurs after c_t and X containing one positive sample h_{t+k} and $N-1$ negative samples h_{j_n} drawn from representations of other samples in the same audio example and the dataset, the following objective is optimised:

$$\mathcal{L}_N = -\sum_k \mathop{\mathbb{E}}_X \left[\log \frac{f_k\left(h_{t+k}, c_t\right)}{\sum_{h_j \in X} f_k\left(h_j, c_t\right)} \right] \tag{2.2}$$

Each encoding pair (h_n, c_t) is evaluated using a scoring function $f(\cdot)$ to estimate how likely a given h_n is the positive sample h_{t+k}. CPC's formulation of the optimal solution for $f(\cdot)$ allows $-\mathcal{L}_n$ to be reformulated as a lower bound on the mutual information of representations $I(h_{t+k}|c_t)$, which also bounds the data $I(x_{t+k}|c_t)$, and is further proven by (Poole et al., 2019). For downstream tasks, both h_t and c_t can be used as representations for new observations x, depending on whether context is helpful for solving it.

Recently, the contribution of mutual information to the success of CPC has been reconsidered: its performance depends on an inductive bias in the choice of a specialised architecture and the parameterisation of the mutual information critic (Tschannen et al., 2020).

2.4.3 SimCLR

SimCLR is a recently proposed contrastive learning technique for learning effective representations of images in a self-supervised manner without relying on specialised architectures and powerful autoregressive modeling (Chen et al., 2020a). The key findings of SimCLR boils down to Yann LeCun's aforementioned 'cake analogy' (2.1): given a large enough neural network, a lot of unlabeled data for pre-training with a pretext task, supervised learning really becomes the icing on the cake of artificial intelligence. Contrary to prior contrastive learning methods (Hénaff et al., 2019; Hjelm et al., 2019; Oord et al., 2019), SimCLR does not require a specialised encoder architecture or powerful autoregressive modeling to learn useful representations.

Instead, it relies on strong data augmentations and very large batch sizes to make the learned representations more rubust and the contrastive pretext task harder. When finetuning a *linear* classifier using the self-supervised learned representations from the pre-trained encoder, it achieved 76.5% top-1 accuracy on ImageNet on the task of image classification. For comparison, the same encoder architecture (ResNet-50) in a standard supervised setting scores 76.6% top-1 accuracy. A next iteration of SimCLR surpassed this supervised benchmark by a significant margin: SimCLRv2 achieves 79.8% top-1 accuracy (Chen et al., 2020b)[5]. [6]

The framework has four core components: 1) a composition of stochastic data augmentations that augment every image into two, correlated versions, 2) a non-linear neural network, 3) a linear or non-linear projection neural network and 4) a contrastive loss function.

The series of data augmentations the authors studied are: random cropping, resizing, horizontal flipping, cutout, color distortion (jitter, hue, dropping), gaussian noise, gaussian blur and sobel filtering. As noted earlier, the first four augmentations are considered spatial transformations, the latter four are spectral transformations. For the sake of simplicity, they used standard ResNet encoders as the encoder neural network and feed forward neural networks for the projection layers. Normalised temperature-scaled cross entropy loss is used as the contrastive loss function, which is shown in equation 2.3. Similarly to InfoNCE, it is a categorical cross entropy loss, but uses a different scoring function f and a temperature-scaling parameter τ.

$$\mathcal{L} = -\log \frac{\exp\left(f\left(z_i, z_j\right)/\tau\right)}{\sum_{k=1}^{N} \mathbb{1}_{[k \neq i]} \exp\left(f\left(z_i, z_k\right)/\tau\right)} \tag{2.3}$$

Batches of $2N$, i.e., every sample has a corresponding, augmented view, are used for pre-training. When training on larger batch sizes, they attribute the increased effectiveness of the learned representations to the increased complexity of the contrastive learning task. Simply put, it makes it harder for the model to infer the positive pair when increasing the pool of negative examples. With a batch size

[5] BYOL surpassed the supervised benchmark a few days before SimCLRv2 was published (Grill et al., 2020), but SimCLRv2 exceeded their scores.

[6] Very recently, BYOL proposed an online and target network model to mitigate the use of many negative samples (Grill et al., 2020). While they contributed interesting new findings, it goes beyond the scope of this thesis.

up to 8192 samples (that is 16384 samples in total during training), learning becomes unstable for standard stochastic gradient descent. To stabalise training, they employ the LARS optimiser (You et al., 2017).

2.5 CNN's for Audio

Most papers in MIR utilise convolutional neural networks (CNN) on audio in the time-frequency domain, i.e., they use CNN's on (mel-)spectograms, CQT's, etc., to learn representations of audio (Böck et al., 2016; Chen and Su, 2019). We omit describing common convolution architectures for these papers because they operate in the time-frequency domain, while our work resides in the time domain. To learn an acoustic model from raw audio signals in the time domain, deep convolutional neural networks have proven to be useful (Lee et al., 2018; Pons et al., 2017). Large receptive fields are often used to mimic the behavior of bandpass filters, while subsequent layers control the model capacity. Auxillary layers, e.g., batch normalisation layers that suppress exploding and vanishing gradients, strong activation functions and residual connections are often deployed between convolution layers to help stabalise training for deep neural networks (He et al., 2016; Ioffe and Szegedy, 2015a). Table 2.3 displays a convolution block incorporating such measures.

Convolution Block

Layer	Output Size (Sequence Length × Channels)
Conv	h_in × h_out
BatchNorm	h_out
ReLU	-

Table 2.3: Convolution block consisting of a parameterised convolution layer and batch normalisation and ReLU activation layers.

2.5.1 SampleCNN

SampleCNN is a model architecture specifically designed for the classification of raw audio signals (Lee et al., 2018). It uses many layers and uses small receptive fields and aggressive pooling modules to obtain a sample-level representation of a signal (Lee et al., 2018). Its architecture is visualised in Table 2.4. The kernel- and striding sizes differ slightly based on the sample rate of the input. As a trade-off between hardware constraints during training, e.g., GPU memory size, a sample rate of 22050 *Hz* yielded the highest results. This configuration's encoder, called SampleCNN 3^9 for its pooling size and number of convolution blocks, is the default encoder used for the ablation experiments in this thesis and is shown in Figure 2.4.

2.6 Music Tagging

Music (auto) tagging, or music classification, is the task of automatically attributing metadata to a fragment of musical audio. The attribution can be a single genre or multiple 'tags' that best describe the contents of the signal. The type of task can thus either be a multi-

SampleCNN 3^9 Model

Layer	Output Size	Parameters		
	(Sequence Length × Channels)	Kernel	Stride	Padding
Input	59049 × 1	3	3	0
ConvBlock	19683 × 128	3	1	1
MaxPool	6561 × 128	3	3	1
ConvBlock	6561 × 128	3	1	1
MaxPool	2187 × 256	3	3	1
ConvBlock	2187 × 256	3	1	1
MaxPool	729 × 256	3	3	1
ConvBlock	729 × 256	3	1	1
MaxPool	243 × 256	3	3	1
ConvBlock	243 × 256	3	1	1
MaxPool	81 × 256	3	3	1
ConvBlock	81 × 256	3	1	1
MaxPool	27 × 256	3	3	1
ConvBlock	27 × 256	3	1	1
MaxPool	9 × 256	3	3	1
ConvBlock	9 × 512	3	1	1
MaxPool	3 × 512	3	3	1
ConvBlock	3 × 512	3	1	1
MaxPool	1 × 512	3	3	1
ConvBlock	1 × 512	3	1	1
Dropout (0.5)	1 × 512	-	-	-
FC	50	-	-	-

Table 2.4: SampleCNN 3^9 Model, with 59049 samples (2678 ms) as input. Each ConvBlock consists of the modules presented in Table 2.3

class or multi-label classification problem. The attributes range from tags that describe the (1) genre, e.g., jazz, pop, metal, (2) moods, e.g., happy, sad, (3) instruments, e.g., guitar, piano, harp, or (4) even more semantic descriptions, e.g., beautiful, of a fragment of music.

The idea behind choosing this downstream task, is that the contrastive learning objective lends itself to such a discriminative task. Moreover, music tags describe many facets of music. When a self-supervised model is able to learn an effective mapping of the versatility of such semantic tasks on high-dimensional signals, it paves the way for more specific musical tasks, e.g., chord recognition. [7]

[7] The initial idea in this thesis was to evaluate more downstream musical tasks under the learned representations by a self-supervised model, but we quickly found that focussing on one task in such an unexplored field was enough, and leave the evaluation of other tasks for future research.

2.6.1 *Evaluation Metrics*

To directly compare the results of this thesis with prior work (Dieleman and Schrauwen, 2014; Lee et al., 2018; Pons et al., 2017), we employ the $\text{ROC} - \text{AUC}_{\text{TAG}}$ and $\text{PR} - \text{AUC}_{\text{TAG}}$ evaluation metrics. The ROC-AUC is the area under the receiver operating characteristic curve that evaluates binary classification problems by summarising the True Positive and False Positive rates. An ROC-AUC score of 1 means the classifier is able to perfectly distinguish true- from false positive labels. When 0, the classifier predicts exactly every true positive as a false positive class and the other way around. With an ROC-AUC of 0.5, the classifier is unable to distinguish between true- and

false positive classes. The ROC-AUC metric suffers over-optimistic scores when classes are inbalanced in the dataset (Davis and Goadrich, 2006). We therefore also employ the PR-AUC evaluation metric, which measures the area under the precision-recall curve. The precision indicates how many correct positive predictions are made. Recall quantifies the number of relevant correct predictions.

As the subscripts in both metrics show, we measure the TAG performance on our task, but also measure the CLIP performance. The evaluation metrics are measured globally for the whole dataset, i.e., for the tag metric we measure the retrieval performance on the tag dimension (column-wise) and for the clip metric we measure the performance on the clip dimension (row-wise). Intuitively, the tag retrieval performance tells us how well the model is able to correctly retrieve all the music fragments (clips), given the tags, while the clip retrieval performance how well the model retrieves all tags given a clip.

Summary

In this chapter, we recapitulated the foundations which are used to build on in this thesis. We gave a brief overview of the different strategies of representation learning to clarify self-supervised learning, i.e., a form of unsupervised learning that formulates the learning objective so that it retrieves a supervisory signal from (transformations of) the data itself. We gave a more extensive description of existing literature on self-supervised learning, important pretext tasks and their strengths, and made an important distinction between learning spectral and spatial relationships when only using a pretext task to learn on data. Subsequently, we addressed self-supervised learning techniques in the audio domain and detailed some of the audio transformation pipelines and augmentations used in the literature. We discussed the contrastive learning paradigm within self-supervised learning, detailing contrastive predictive coding, SimCLR, and noise-contrastive estimation losses. We discussed encoders used in the literature on deep learning on audio in the time domain, and described the SampleCNN model which is used in this thesis. Finally, we gave a description of the evalation metrics we used to evaluate the performance of our proposed self-supervised model on raw audio: CLMR.

3 CLMR

In this chapter, the core components of the CLMR framework are detailed. The data augmentation pipeline is outlined, as well as the evaluation procedure of the representations learned by CLMR. Concluding, the transfer learning experiments are explained.

The following core components of the framework are outlined in the following sections:

- A stochastic composition of data augmentations that produces two correlated, augmented examples of the same audio segment, the 'positive pair', denoted as x_i and x_j. This is done for all segments in the mini-batch, resulting in $2N$ augmented examples per mini-batch.
- An encoder neural network $g_{\text{enc}}(\cdot)$ that encodes the augmented examples to their latent representations.
- A projector neural network $g_{\text{proj}}(\cdot)$ that maps the encoded representations to the latent space where the contrastive loss is formulated.
- A contrastive loss function, which aims to identify x_j from the negative examples in the mini-batch $\{x_{k\neq i}\}$ for a given x_i.

The complete framework is visualised in Figure 3.1.

3.1 Data Augmentations

We designed a chain of augmentations for raw audio waveforms to make it harder for the model to identify the correct pair of examples. The following augmentations were applied on x_i and x_j independently:

1. A random segment of size N is selected from a full piece of audio, without trimming silence (e.g., the intro or outro of a song). The independently chosen segments for x_i and x_j could overlap or be very disjoint, allowing the model to infer both local and global structures. This intuition is visualised in Figure 3.2.
2. The polarity of the audio signal is inverted, i.e., the amplitude is multiplied by -1, with probability p_{invert}.
3. Additive White Gaussian Noise is added with a high signal-to-noise ratio to the original signal with probability p_{noise}.
4. The gain is reduced between $[-6, 0]$ decibels with probability p_{gain}.
5. A filter is applied with probability p_{filter}. A coin flip determines whether it is a low-pass or a high-pass filter. The cut-off frequencies are randomly drawn from the uniform distribution $[2200, 4000]$ or $[200, 1200]$ respectively.

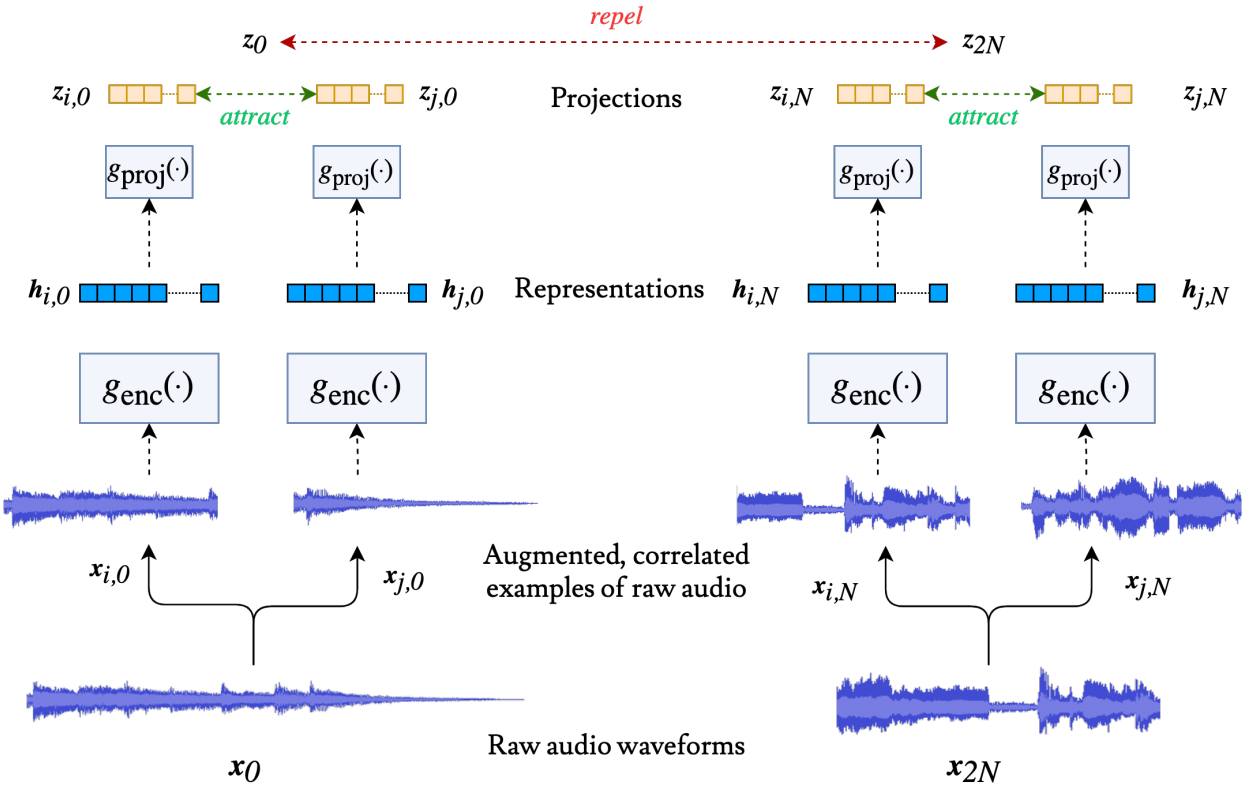


Figure 3.1: The CLMR framework operating on raw audio, in which the contrastive learning objective is directly formulated in the latent space of correlated, augmented examples of pairs of raw audio waveforms.

6. The signal is delayed with probability p_{delay}. The delay time is randomly chosen from values between 200-500ms, with 50ms increments. The volume factor of the delayed signal that is added to the original signal is 0.5.
7. The signal is pitch shifted with probability p_{pitch}. The pitch transposition interval is drawn from a uniform distribution consisting of intervals ranging from a fifth below to a fifth above the original signal's scale.
8. Reverb is added to the signal with probability p_{reverb}. The impulse response's room size, reverbation and damping factor is randomly chosen from a uniform distribution of values between $[0 - 100]$.

The space of augmentations is not limited to these operations and could easily be extended to, e.g., randomly applying chorus, distortion and other modulations, as outlined in Section 2.3.2.

3.2 *Mini-Batch Composition*

We sample one song from the mini-batch, augment it into two examples, and treat them as the positive pair. We treated the remaining $2(N-1)$ examples in the mini-batch as negative examples, and did not sample the negative examples explicitly. A larger batch size makes the model's objective harder – there are simply more negative samples the anchor sample needs to identify the positve samle from – but it can substantially improve model performance (Chen

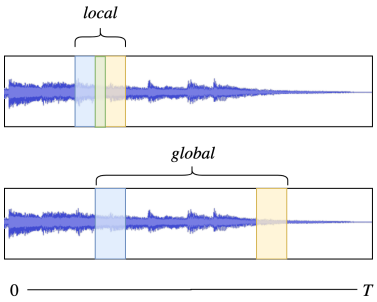


Figure 3.2: During pre-training, a random segment of size N is selected from a full piece of audio. The independently chosen segments x_i (blue) and x_j (yellow) could overlap or be disjoint, which should allow the model to infer both local and global structures.

et al., 2020a). This introduces a practical problem for raw audio when training on a GPU, as the input dimensionality of a raw waveform is higher for high sample rates. The batch size can be increased more easily when audio is re-sampled at lower sampling rates: the number of examples the model is exposed to at once can be higher when the number of audio samples is lower.

Alternatively, multiple GPU's can be used for training, but this introduces another practical problem: batch normalisation (Ioffe and Szegedy, 2015b) is used in the encoder to stabilise training. When training in a distributed, parallel manner, the batch normalisation statistics (mean/variance) are usually aggregated locally per device. Positive examples are sampled on the same device, leading to potential leakage of batch statistics which improves training loss, but counteracts learning of useful representations. We used global batch normalisation, which aggregates the batch statistics over all devices during parallel training, to alleviate this issue. We leave the effect of different stabilisation strategies, e.g., layer normalisation (Hénaff et al., 2019), for future work.

3.3 *Encoder*

To directly compare a state-of-the-art end-to-end supervised model against a self-supervised model operating on raw waveforms, we use the SampleCNN model as our encoder (Lee et al., 2018). Similar to the supervised approaches, we use an audio input of 59 049 samples for audio with a sample rate of 22 050 Hz. In this configuration, the SampleCNN encoder g_{enc} consists of 11 blocks, each with a convolutional layer with a filter size of 3, batch normalisation, ReLU activation and max pooling with pool size 3. The fully connected and dropout layers are removed, yielding a 512-dimensional feature vector for every sample of audio. This feature vector is subsequently mapped to a different latent space by the projector network g_{proj} where the contrastive loss function is defined. We adjust the audio input length and the encoder's blocks according to the configurations proposed in (Lee et al., 2018) when training on audio sampled at different sampling rates (16 000, 12 000 and 8 000 Hz).

We found working with a batch size of 48, i.e., 96 samples per batch since we use $2N$ samples for our negative sampling strategy, and the 3^9-SampleCNN encoder configuration to work well and easier to compare with against related work using supervised methods (Dieleman and Schrauwen, 2014; Lee et al., 2018; Pons et al., 2017). We leave model scaling for future work.

3.4 *Projector*

The feature vectors from the encoder can be directly used in the learning objective, but SimCLR shows that formulating the objective on encodings mapped to a different latent space by a parameterised function helps the effectiveness of the representations (Chen et al.,

2020a). We evaluate the performance improvement when using a linear layer $z_i = Wh_i$, non-linear layer $z_i = W^{(2)}\,\mathrm{ReLU}(W^{(1)}h_i)$ and an identity function $z_i = h_i$ as the projector.

3.5 *Contrastive Loss Function*

In keeping with recent findings on several objective functions in contrastive learning (Chen et al., 2020a), the contrastive loss function used in this model is normalised temperature-scaled cross-entropy loss, commonly denoted as *NT-Xent loss*:

$$\ell_{i,j} = -\log \frac{\exp\left(\mathrm{sim}\left(z_i, z_j\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp\left(\mathrm{sim}\left(z_i, z_k\right)/\tau\right)} \tag{3.1}$$

Instead of using a scoring function that preserves the mutual information between vectors, the pairwise similarity is measured using cosine similarity (sim). It introduces a new temperature parameter τ to help the model learn from hard negatives. The indicator function $\mathbb{1}_{[k\neq i]}$ evaluates to 1 iff $k \neq i$. This loss is computed for all pairs, both (z_i, z_j) and (z_j, z_i), resulting in the following total loss function:

$$\mathcal{L} = \frac{1}{2N} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbb{1}_{[i\neq j]} \ell_{i,j} \tag{3.2}$$

3.6 *Evaluation*

The evaluation of representations learned by self-supervised models is commonly done with linear evaluation (Chen et al., 2020a; Hjelm et al., 2019; Oord et al., 2019), which measures how linearly separable the relevant classes are under the learned representations. We obtain representations h_t for all data points X from a frozen CLMR network after pre-training has converged, and train a linear classifier using these self-supervised representations on the downstream task of music classification. For CPC, the representations are extracted from the autoregressor, yielding context vector c of 256 dimensions, which is global-average pooled to obtain a single vector of 512 dimensions. For CLMR, the representations h from the encoder are used instead of the representations z from the projector.

3.7 *Transfer Learning*

To test the generalisability of the learned representations, we also pre-trained CLMR on different datasets than those we use for fine-tuning. We pre-train CLMR on the Million Song Dataset, freeze the weights of the network, and subsequently process all datapoints X from the smaller MagnaTagATune dataset to obtain representations h, on which we perform the same linear evaluation procedure outlined in the previous paragraph.

4 Datasets

In this chapter, we first give a short background on the standardisation of datasets and evaluation procedures in the field of MIR. Then, we give a more detailed description of the datasets that were used for the experiments in this thesis.

We used the MagnaTagATune dataset and Million Song Dataset (Bertin-Mahieux et al., 2011) for pre-training and evaluation.

For the transfer learning experiments, we pre-train CLMR on the Million Song Dataset, fault-filtered GTZAN (Sturm, 2013; Tzanetakis and Cook, 2002), McGill Billboard (Burgoyne et al., 2011) and Free Music Archive (Defferrard et al., 2017) datasets. We subsequently perform linear evaluation of the self-supervised learned representations on the MagnaTagATune dataset.

4.1 MIREX

There are several benchmark datasets that are used to evaluate music classification algorithms with. One of the first attempts to unify datasets and evaluation procedures for music (classification) algorithms is MIREX: the Music Information Retrieval Evaluation eXchange. This 'exchange' was started as a platform to evaluate newly published algorithms on many tasks in the field of MIR. The MIR tasks range from chord recognition, music key detection, audio fingerprinting to music classification. Along with the unification of evaluation procedures, it has also produced standard datasets to benchmark algorithms with. For the task of music classification, the MagnaTagATune dataset (Law et al., 2009) is often used. For chord recognition, the Billboard dataset is regarded as a standard dataset (Burgoyne et al., 2011).

4.2 MagnaTagATune Dataset

The MagnaTagATune dataset was compiled by crowdsourcing tags from a game called 'TagATune' using music from the Magnatune label. For the MagnaTagATune dataset, we used the original, MIREX 2009 version, consisting of 25 863 songs, and the same dataset split, so that we can compare our results with previous work easily (Dieleman and Schrauwen, 2013; Lee et al., 2018; Pons et al., 2017). It should be noted that this original version contains tag labels that are synonymous, e.g., 'female', 'woman', 'no vocal', 'no voice' and also contains tracks that do not have any labels. The top-50 tags in the MagnaTagATune dataset are shown in Table 4.1.

The distribution of the number of fragments per class is skewed: there are more fragments containing guitar and classical tag attributes than 'country' or 'harp' attributes. A possible consequence of this

guitar	classical	slow
techno	strings	drums
electronic	rock	fast
piano	ambient	beat
violin	vocal	synth
female	indian	opera
male	singing	vocals
no vocals	harpsichord	loud
quiet	flute	woman
male vocal	no vocal	pop
soft	sitar	solo
man	classic	choir
voice	new age	dance
male voice	female vocal	beats
harp	cello	no voice
weird	country	metal
female voice	choral	

Table 4.1: 50 most popular tags in the MagnaTagATune dataset

class imbalance, is that CLMR learns representations that separate, e.g., 'guitar' music and 'flute' music well, but it is harder to learn representations that separate 'country' music from 'choral' music.

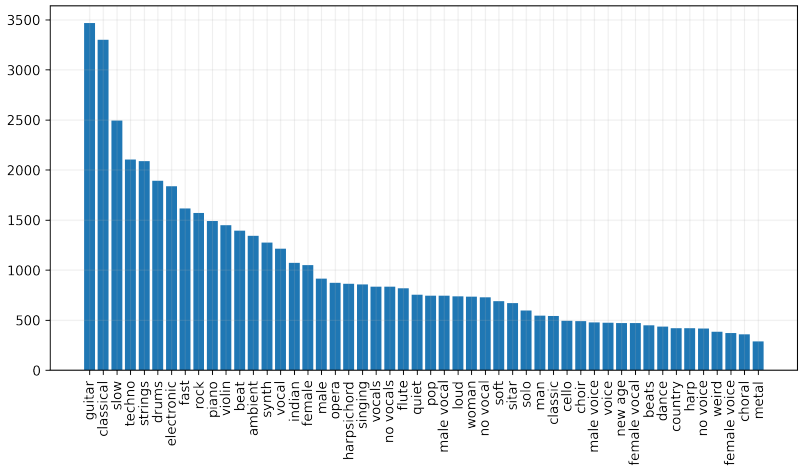


Figure 4.1: Distribution of tags of the MagnaTagATune dataset

4.3 *Million Song Dataset*

The Million Song Dataset consists of 1 million audio features and metadata of contemporary pop songs. It is commonly used for benchmarking on a larger scale. These features were compiled by The Echo Nest, a music data company that has since been purchased by Spotify. While they provide the audio features that are computed using their proprietary algorithms, the raw audio data is not provided due to the infringement of copyright when published freely and publicly.

Since this thesis uses raw audio to train and evaluate the CLMR model on, it was a challenge to obtain the raw audio of the contempory pop songs. Originally, it could be obtained by accessing

30-second fragments using the internal IDs matched with those from the '7 digital' music service. [1] Since it does not provide this service anymore, we had to obtain the dataset from a another MIR research group that archived the 7digital fragments. [2]

[1] 7digital.com

[2] We would like to thank Jongpil Lee from the Korea Advanced Institute of Science and Technology for providing us access to their server to retrieve the 30-second raw audio fragments.

The Million Song Dataset's tags were compiled by cross-referencing it with the crowdsourced Last.fm dataset (Bertin-Mahieux et al., 2011). Similar to the MagnaTagATune dataset, we use only tracks that were annotated with tags from the set of top-50 most popular tags. This results in 241,904 unique songs.

The tags for the Million Song Dataset are more overlapping, e.g., 'rock' and 'classic rock', and contain more semantic tags, e.g., 'beautiful', 'happy' and 'sad', which are arguably harder to linearly separate when fine-tuning the linear classifier. The top-50 tags are shown in Table 4.2. Similar to the MagnaTagATune dataset, the classes are also imbalanced in this dataset. The 'rock' attribute is overrepresented in the dataset. While we did not correct for this imbalance, we reckon it has a similar effect on the learned representations as described earlier for the MagnaTagATune dataset.

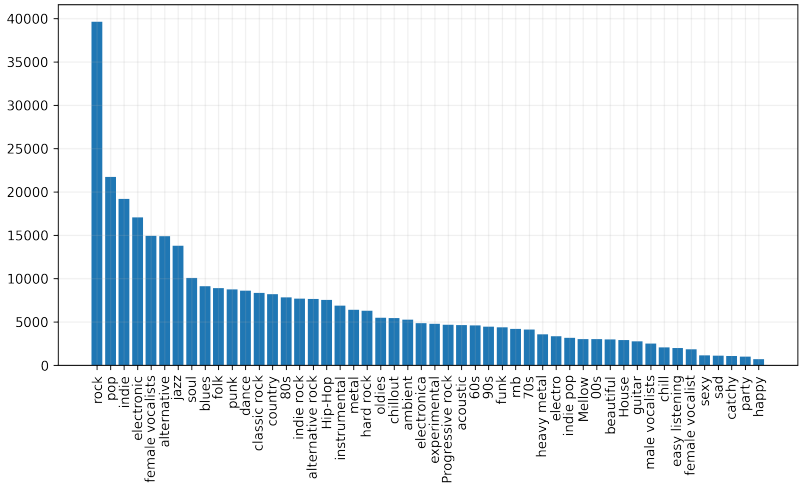


Figure 4.2: Distribution of tags of the Million Song Dataset

4.4 *Transfer Learning Datasets*

4.4.1 *McGill Billboard Dataset*

From the McGill Billboard dataset, we use 461 audio files of contemporary pop songs for training. While this dataset is most often used for evaluating chord recognition algorithms, we use the audio solely for self-supervised pre-training. The audio is only used for self-supervised pre-training.

4.4.2 *Free Music Archive*

Similarly, we use the Free Music Archive dataset, consisting of 22 413 unique multi-labeled songs for the 'medium' version,

rock	pop	alternative
indie	electronic	female vocalists
dance	00s	alternative rock
jazz	beautiful	metal
chillout	male vocalists	classic rock
soul	indie rock	mellow
electronica	80s	folk
90s	chill	instrumental
punk	oldies	blues
hard rock	ambient	acoustic
experimental	female vocalist	guitar
hip-hop	70s	party
country	easy listening	sexy
catchy	funk	electro
heavy metal	progressive rock	60s
rnb	indie pop	sad
house	happy	

Table 4.2: 50 most popular tags in the Million Song Dataset

4.4.3 GTZAN

The fault-filtered GTZAN dataset contains 930 segments of 30 seconds, each having a single label denoting its genre (McFee et al., 2015; Tzanetakis and Cook, 2002). The dataset is made up of 10 genres: classical, country, disco, hip-hop, jazz, rock, blues, reggae, pop and metal. The audio is again only used for self-supervised pretraining.

5 Implementation

In this chapter, we give a comprehensive overview of the implementation of the CLMR model. We provide a PyTorch (Paszke et al., 2019) implementation for both CLMR and CPC. The code implementation can be found on GitHub.[1]

[1] https://github.com/spijkervet/CLMR

5.1 CLMR

The CLMR model extends the vision contrastive learning framework, SimCLR (Chen et al., 2020a). We implemented their paper in PyTorch and ran extensive experiments to ensure it met the original paper's results. We provide the code on GitHub as well. [2] [3]

[2] With 250+ stars and 50+ forks, our code has grown into one of the most popular PyTorch implementations of this framework: https://github.com/spijkervet/SimCLR

[3] After we published the code, the original author, Ting Chen, also published their implementation in TensorFlow. We are cited as one of the PyTorch implementations in their work https://github.com/google-research/SimCLR

5.1.1 Optimising Audio Transformations

The data augmentation pipeline consists of functions from the following code libraries:

```
essentia
torchaudio
librosa
sox
wavaugment
```

We use large mini-batch sizes for training, and while every mini-batch must contain randomly augmented examples, it is of significant importance to optimise the runtime of a parallelised augmentation pipeline. Since audio transformations are CPU-intensive operations, most libraries have optimised their code by creating an interface between Python and languages that map their code more efficiently to machine instructions, e.g., the C language, to avoid a bottleneck on these augmentations. [4] However, both pitch-shifting and reverberation have not yet been fully optimised for Python. The code implementation of WavAugment provides a Python - C++ interface to interact with all audio effects in the sox library from within Python (Kharitonov et al., 2020). We use this implementation to significantly speed up pitch-shift and reverberation transformations in our augmentation pipeline.

[4] Python is an interpreted language, which makes it slower than machine code because making interpretations of instructions takes longer than executing machine instructions directly.

5.1.2 GPU Parallelisation

PyTorch provides two interfaces to parallelise training on GPU's. This can be used to either scale up the model, i.e., by increasing the number of parameters or by increasing the mini-batch size, or to speed up training. The 'DataParallel' (DP) module parallelises the data across multiple GPU's on a single node, while 'DistributedDataParallel' (DDP) distributes it over multiple GPU's across multiple nodes. We use either DP or DDP to speed up training or to increase the mini-batch size. The maximum available hardware provided for

this thesis was a a single 4 × Titan RTX node with 96 gigabytes of GDDR6 memory. [5] This allowed us to train our largest model with a mini-batch size of 456. We leave further model scaling for future work.

[5] We would like to thank SURFsara for providing the GPU nodes on the lisa system.

As described earlier, batch normalisation is used in the convolution block to stabalise training, especially for larger mini-batch sizes 2.3. Since information about batch statistics could leak to the learning objective, we utilise global batch normalisation. It aggregates the operation from the devices to a single GPU device, and subsequently distributes the results to the other devices.

For multi-node training using DDP, the losses from all devices need to be aggregated in a similar way to avoid leakage of mini-batch information, i.e., the positive, anchor and negative samples. We used a gathering operation in PyTorch to aggregate losses from the NT-Xent function from all GPU devices.

5.1.3 Encoder

Our code implementation for CLMR allows for any encoder to be attached. In this thesis, we implemented the SampleCNN encoder in PyTorch as our feature extractor. The SampleCNN encoder consists of 11 convolution blocks, with varying kernel sizes and strides depending on the sample rate of the input audio. Its full structure is already shown in Table 2.4. With an input audio length of 2.7 seconds, the configuration of the kernel size and strides are listed in Table 5.1. The number of channels in the convolution block for each configuration is kept constant: $[128, 128, 128, 128, 256, 256, 256, 256, 256, 512, 512]$.

Sample rate (Hz)	Audio length	Kernel size / Stride
22050	59049	$[3, 3, 3, 3, 3, 3, 3, 3, 3]$
16000	43470	$[3, 3, 3, 3, 3, 3, 5, 2, 2]$
8000	20736	$[3, 3, 3, 2, 2, 4, 4, 2, 2]$

Table 5.1: Kernel size and strides for different sample rates

5.2 Contrastive Predictive Coding

We refer the reader to Figure 2.2 for a schematic overview of CPC, which puts the following details in a better perspective.

We adjusted the original CPC encoder g_{enc} to a structure similar to that of SampleCNN's to compare the effectiveness of this contrastive learning strategy more directly with CLMR and supervised benchmarks. The encoder g_{enc} consists of 7 layers with 512 filters each, and filter sizes $[10, 6, 4, 4, 4, 2, 2]$ and strides $[5, 3, 2, 2, 2, 2, 2]$. This results in a downsampling factor of 490, which yields a feature vector for every ≈ 5 ms of audio for an input of 59 049 samples. Instead of relying on max-pooling, the filter sizes and strides are adjusted accordingly to parameterise and facilitate downsampling. We also increased the number of prediction steps k to 20, effectively asking the network to predict 100 ms of audio into the future. The mini-batch size, i.e., the number of training examples the model is exposed to at once, is

set to 64 from which 15 negative samples in the contrastive loss are drawn.

5.3 Optimisation

For asymmetric, non-linear activation functions like ReLU, it has been demonstrated that initialising the model parameters using Kaiming initialisation allows for faster model convergence (He et al., 2015). We employ Kaiming initialisation for both CLMR and CPC. The following optimisers are used for pre-training. For smaller batch sizes, i.e., < 96, we use the Adam optimiser with a learning rate of 0.0003 and $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For larger batch sizes, i.e., ≥ 96, we use the LARS optimiser with square root learning rate scaling shown in Equation 5.1, a cosine annealing schedule 5.1 for the learning rate and a weight decay of 10^{-6}. This has shown to benefit contrastive learning when using a mini-batch size ≤ 4096 (Chen et al., 2020a).

$$LearningRate = 0.075 * \sqrt{BatchSize} \tag{5.1}$$

For linear evaluation, we use the Adam optimiser with a learning rate of 0.0003 and a weight decay of 10^{-6}. Backpropagation is only done in the fine-tune head, i.e., it only optimises the linear layer or MLP. The pre-trained encoder remains frozen in all evaluation procedures, for the full classification, efficient classification and transfer learning experiments. We also employ an early stopping mechanism when the validation scores do not improve for 3 epochs. [6]

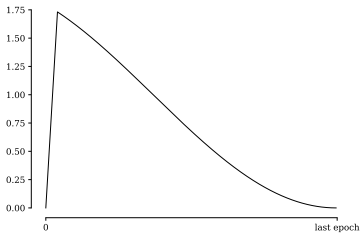


Figure 5.1: Learning rate schedule: a warm-up is performed before the learning rate is adjusted using a cosine annealing schedule. In this example, the learning rate linearly scales to 1.75 before decreasing back to near-zero at the last epoch.

[6] Early stopping stops training on the larger train set, when it has shown to generalise on a smaller validation set.

6 Experimental Results

In this chapter, we outline the results obtained from the experiments and detail the outcomes of the ablation study. The quality of our models' representations are evaluated using the music classification task.

Model	Dataset	ROC-AUC$_{\text{TAG}}$	PR-AUC$_{\text{TAG}}$
CLMR (ours)	MTAT	88.49 (**89.25**)	**35.37** (**35.89**)
Pons et al.†	MTAT	89.05	34.92
SampleCNN†	MTAT	88.56	34.38
CPC (ours)	MTAT	86.60 (87.99)	30.98 (33.04)
1D CNN†	MTAT	85.58	29.59
Pons et al.†	MSD	87.41	**28.53**
SampleCNN†	MSD	**88.42**	-
CLMR (ours)	MSD	85.66	24.98

Table 6.1: Tag prediction performance on the MagnaTagATune (MTAT) dataset and Million Song Dataset (MSD), compared with fully supervised models† trained on raw audio waveforms. We omit works that operate on audio in the time-frequency domain. For the supervised models, the tag-wise scores are obtained by end-to-end training. For the self-supervised models, the scores are obtained by training a *linear*, logistic regression classifier using the representations from self-supervised pre-training. Scores in parenthesis show performance when adding one hidden layer to the logistic regression classifier, making it a simple multi-layer perceptron.

6.1 Quantitative Evaluation

The most important goal set out in this thesis, is to evaluate the difference in performance between a fully supervised and a self-supervised objective when learning representations, using the exact same encoder set-up. The CLMR model in our experiments uses the SampleCNN encoder network. When SampleCNN is trained in a fully supervised manner, it reaches an PR-AUC score of 34.92. CLMR exceeds this supervised benchmark with a PR-AUC of 35.37, despite task-agnostic, self-supervised pre-training and a *linear* classifier for fine-tuning. An additional 0.5 PR-AUC performance gain is added by adding one extra hidden layer to the classifier. Evaluation scores of the best-performing CLMR, CPC and other wave-form based models are shown in Table 6.1.

CLMR also outperforms the current state-of-the-art waveform-based model in the task of automatic music tagging (Pons et al., 2017) in both evaluation metrics for the MagnaTagATune dataset.

The performance on the Million Song Dataset is lower than that of SampleCNN. The highest evaluation scores for the MagnaTagATune dataset are obtained after very long pre-training (10 000 epochs), of which the details are outlined in Section 6.6. While this is feasible for a dataset of the size of MagnaTagATune's, we did not have the equipment available to run the experiment for so long on the Million Song Dataset. Additionally, we attribute the difference in performance to the more semantically complex tags in the Million Song Dataset, e.g., 'catchy', 'sexy', 'happy', or more similar tags, e.g., 'progressive rock', 'clsasic rock' and 'indie rock', which may not be linearly separable.

CPC also shows competitive performance with fully supervised models in the music classification task, despite being pre-trained without

ground truth and using a simple, linear classifier for evaluation. Despite CPC's good performance, self-supervised training indeed does not require a memory bank or more complex loss functions, e.g., those incorporating mutual information or more explicit negative sampling strategies, to learn useful representations.

6.2 *Qualitative Analysis*

For a qualitative view of the representations, we show how cleanly they are separable using a *t*-SNE manifold in Figure 6.1. Figure 6.2 shows in more detail that the difference in performance between self-supervised and supervised models is marginal: there is no single tag performance difference larger than 4% ROC-AUC, and for practical purposes, CLMR and CPC retrieve tags practically identically to supervised models.

Figure 6.2 also shows that it is especially hard for the self-supervised models to distinguish more semantically complex tags, e.g., 'weird', 'new age', some instrumental tags, e.g., 'stiar', 'female/male voice', or the absence of a vocal, i.e., 'no vocal'. Similarly to some of the tags in the Million Song Dataset, these may be harder to linearly separate.

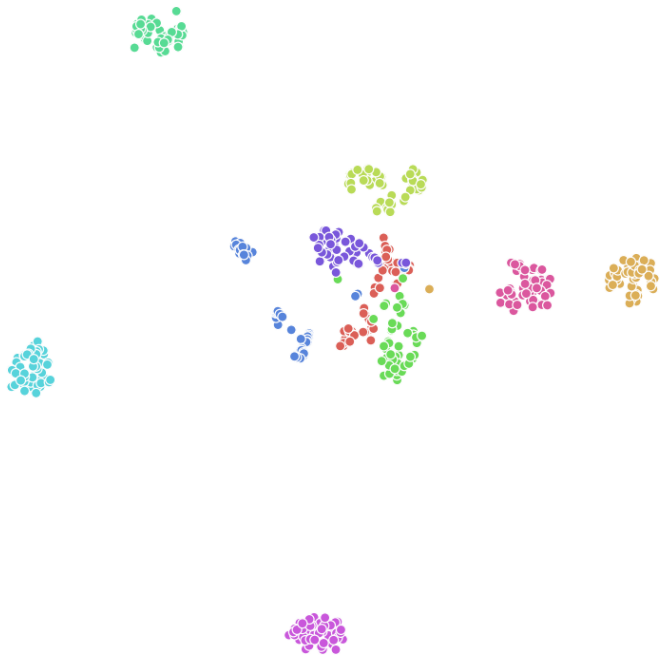


Figure 6.1: *t*-SNE manifold visualisation from audio representations learned by a converged CLMR model of a subset of 10 tracks with each 60 segments. Every color represents a separate track.

6.3 *Data Augmentations*

The CLMR model relies on a pipeline of strong data augmentations to facilitate the learning of representations that are more robust and allow for better generalisation in the downstream task. In Figure 6.3, we show the PR-AUC linear evaluation score that is achieved when taking a random slice of audio ('random cropping') and performing one additional, individual augmentation. While all datasets

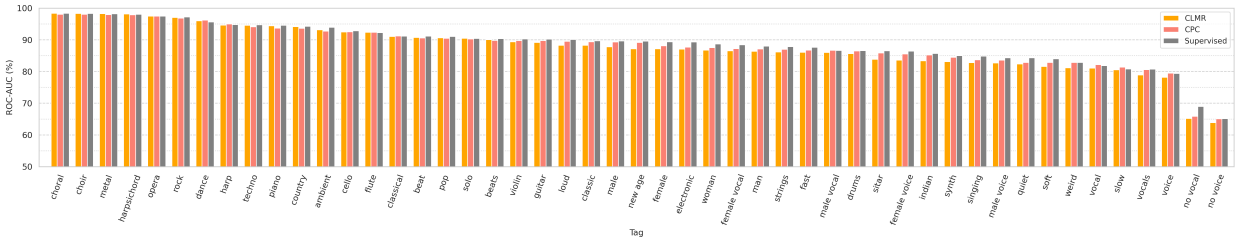


Figure 6.2: Tag-wise ROC-AUC scores for the top-50 tags in the MagnaTagATune dataset, reported for linear, logistic regression classifiers trained on representations of self-supervised models CLMR and CPC, and compared to a fully supervised, end-to-end SampleCNN model.

contain songs of variable length, we always sample a random slice of audio of the same size before applying other augmentations. Since we always have to take a random slice, it makes it harder to assess the individual contribution of each augmentation to the downstream task performance. We therefore consider an asymmetric data transformation setting: we only apply the augmentation pipeline to one branch of the framework, while we settle with an identity function for the other branch (i.e., $t(x_j) = x_j$) (Chen et al., 2020a). The transformations on the augmentation branch are applied with probability $p_t = 1$, i.e., the transformation is always applied.

When only taking a random slice of audio (i.e., a 'crop'), we achieve a PR-AUC score of 30.5. Most augmentations show a similar increase in performance of ±31.5, while adding gain or delay does not impact performance as much. Adding a filter to the pipeline increases the downstream performance significantly.

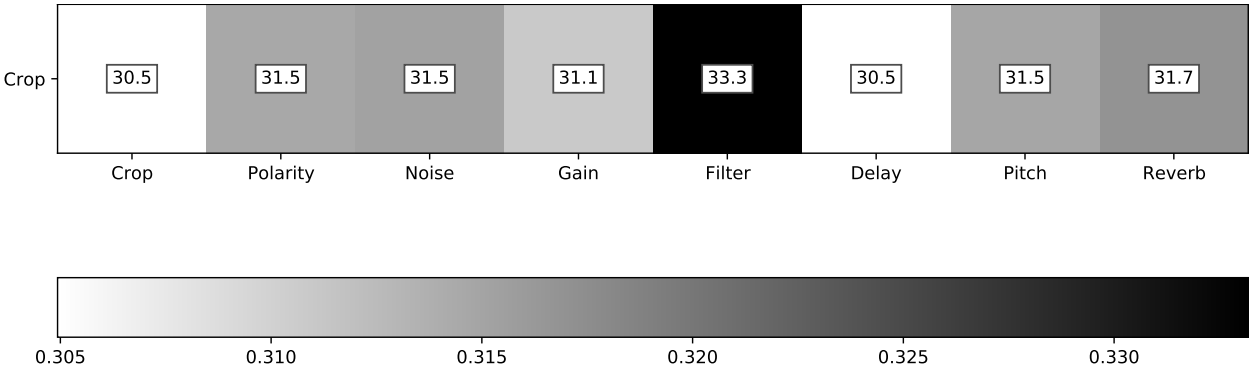


Figure 6.3: The achieved $\mathrm{PR-AUC}_{\mathrm{TAG}}$ score using a random crop together with one other transformation

Besides evaluating the individual contribution of each augmentation with a probability of $p_t = 1$, we also vary this probability: $p_t \in \{0, 0.4, 0.8\}$. This is done to assess the optimal amount of augmentation to each example, i.e., the contrastive learning task should not be too hard, neither too simple, for learning effective representations in the downstream music classification task. The linear evaluation PR-AUC score is shown for each augmentation under a different probability p_t in Figure 6.4. For the Polarity and Filter transformations, performing them more often with a probability of $p_t = 0.8$ is beneficial. For the Delay, Pitch and Reverb transformations, a transformation probability of $p_t = 0.4$ works better than performing them more aggressively.

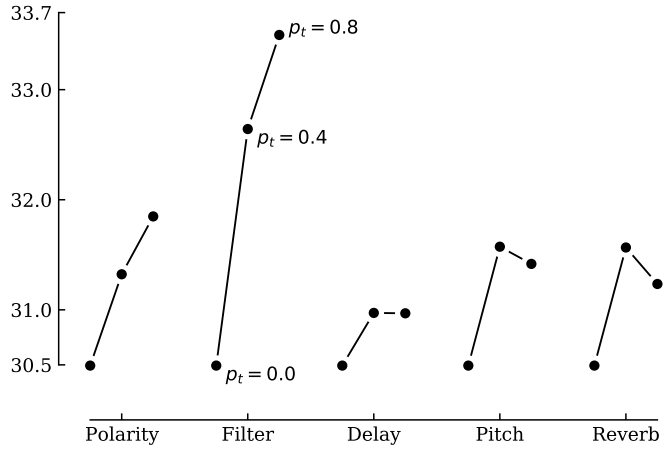


Figure 6.4: $\mathrm{PR} - \mathrm{AUC}_{\mathrm{TAG}}$ scores for transformations under different, consecutive probabilities $p \in \{0.0, 0.4, 0.8\}$

6.4 *Efficient Classification Experiments*

When training on a specific task like music classification, only a limited amount of labeled data may be available. To test the efficient classification capability of the CLMR model, we fine-tune the linear classifier on 1% of the labels in the dataset and report its performance. During the task-agnostic, self-supervised pre-training phase, 100% of the data is used. As outlined previously, the representations that are learned during this phase are subsequently used during linear evaluation.

Figure 6.5 and 6.6 show the PR-AUC scores obtained when increasing the amount of labels available during fine-tuning. For both datasets, fine-tuning using just 1% of the labels yields a large performance difference compared to training in a fully supervised manner, while using the same amount of labels.

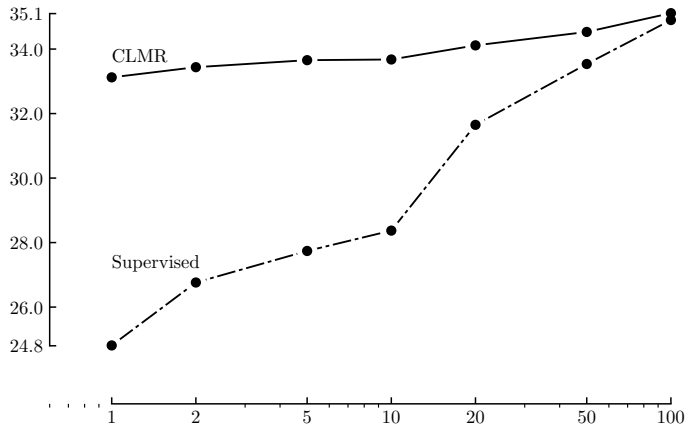


Figure 6.5: Percentage of labels used for training vs. the achieved $\mathrm{PR} - \mathrm{AUC}_{\mathrm{TAG}}$ score on the MagnaTagATune dataset

Using 100× fewer labels, CLMR scores 33.1% PR-AUC compared to 24.8% PR-AUC obtained with an equivalent, end-to-end trained supervised model. Pre-training using a self-supervised objective without labels therefore substantially improves efficient classification: only 1% of the labels are required while maintaining a similar performance.

For the Million Song Dataset, a fully supervised end-to-end trained model exceeds CLMR at 10% of the labels, which are 24,190 unique songs in total. Beyond this amount of music, CLMR is unable to perform in a trend similar to the supervised model.

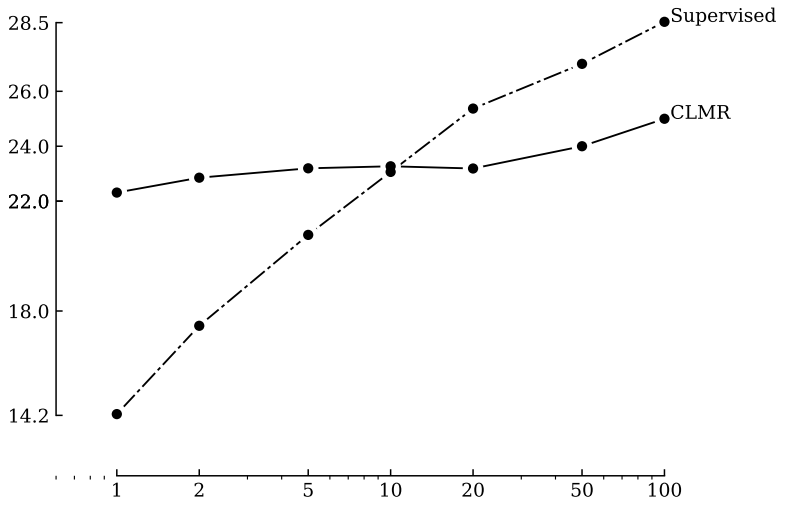


Figure 6.6: Percentage of labels used for training vs. the achieved $PR-AUC_{TAG}$ score on the Million Song Dataset.

6.5 *Transfer Learning Experiments*

The results of the transfer learning experiments are shown in Table 6.2. Both CPC and CLMR show the ability to learn effective representations from datasets different from the evaluation dataset without ground truth, and even exceed accuracy scores of previous, supervised end-to-end systems on raw audio (Dieleman and Schrauwen, 2014). Moreover, both models demonstrate the ability to learn useful representations on the much smaller GTZAN and Billboard datasets. The CLMR model performs better when it is pre-trained on larger datasets, which is expected as it heavily relies on the number of unique, independent examples that make the contrastive learning task harder, resulting in more robust representations. When pre-training on smaller datasets, the autoregressive modelling in CPC can find more useful representations for downstream tasks.

Model	Train Dataset	Eval. Dataset	ROC-AUC	PR-AUC
CLMR	MSD	MTAT	86.57	32.04
CPC	FMA	MTAT	86.34 (87.79)	30.71 (32.47)
CLMR	FMA	MTAT	86.22 (86.63)	30.58 (31.22)
CPC	Billboard	MTAT	85.78 (86.25)	29.68 (30.15)
CPC	GTZAN	MTAT	83.44 (86.06)	26.88 (29.72)
CLMR	Billboard	MTAT	82.73 (84.22)	26.86 (27.82)
CLMR	GTZAN	MTAT	81.88 (85.43)	26.18 (29.49)

Table 6.2: Performance of the self-supervised models when pre-trained on datasets different from the evaluation dataset, again using a linear classifier to evaluate.

6.6 Additional Experiments

6.6.1 Mini-batch Size

The complexity of the pretext task increases with larger mini-batch sizes. To reiterate: in our pretext task, it is harder for the the model to infer the positive pair when increasing the pool of negative examples. To further study the quality of the repesentations for the downstream task performance, given the increase of complexity of the pretext task, we experimented with varying mini-batch sizes while keeping all other parameters the same.

We use the default sample rate of 22 050 Hz, the 3^9 SampleCNN encoder network, train all experiments for 3 000 epochs and set the transformation probabilities to their optimal value, as demonstrated in Section 6.3. All models with varying mini-batch sizes are trained from scratch and evaluated using the linear evaluation procudure outlined in Section 3.6.

While our smallest model already shows competitive performance compared to fully supervised models, the performance increased when using 96 examples per mini-batch. Our largest model did not perform as expected, and scores consistently lower than our middle-sized model. This may be attributed to a sub-optimal configuration of the optimiser, i.e., the LARS optimiser that is used for larger mini-batch sizes. Another theory, is that the task of inferring the positive pair of 2.6 second long audio fragments, in a pool of 912 negative samples, may require even longer training, or is simply too hard. The results of this experiment are shown in Table 6.3.

Mini-batch Size	ROC-AUC$_{\text{TAG}}$	PR-AUC$_{\text{TAG}}$	ROC-AUC$_{\text{CLIP}}$	PR-AUC$_{\text{CLIP}}$
456	88.13	34.87	92.96	68.90
96	88.49	35.11	93.07	69.20
48	87.91	34.56	92.88	68.75

Table 6.3: Effect of the mini-batch size used during self-supervised training on the music classification task performance.

6.6.2 *Training Duration*

Contrastive learning techniques have shown to benefit from longer training compared to their supervised equivalent (Chen et al., 2020a). While larger mini-batch sizes increase the number of negative examples the model can 'see' simultaneously, training for a longer time increases the number of negative examples the model sees overall. We vary the number of epochs we train the model for, while we keep all other parameters constant, i.e., identically to those described in Section 6.6.1. The mini-batch size is set to 96 and all models are again trained from scratch and evaluated using the linear evaluation procedure.

Increasing the self-supervised training duration improves the downstream task performance. Our best results are obtained when training the CLMR model for 10 000 epochs. When training the logistic, linear regression classifier on the learned representations, it perfroms better than a fully, end-to-end trained supervised model. The results of this experiment are shown in Table 6.4.

Epochs	ROC-AUC$_{\text{TAG}}$	PR-AUC$_{\text{TAG}}$	ROC-AUC$_{\text{CLIP}}$	PR-AUC$_{\text{CLIP}}$
10 000	88.47 (89.25)	35.37 (35.89)	93.16 (93.48)	69.32 (70.03)
3 000	88.49 (88.94)	35.11 (35.46)	93.07 (93.27)	69.20 (69.74)
1 000	88.31 (88.64)	34.40 (34.86)	92.89 (93.08)	68.59 (69.15)

Table 6.4: Effect of the self-supervised training duration on the music classification task performance.

6.6.3 *Sample Rates*

When re-sampling the audio to 8 000 Hz and 16 000 Hz respectively, there is a marginal penalty to the final scores for the self-supervised models, which is in line with previous work (Lee et al., 2018). In Table 6.5, we show all linear evaluation scores when no additional transformations are performed (i.e., only random cropping) to isolate the contribution of each individual sample rate.

Sample rate	ROC-AUC$_{\text{TAG}}$	PR-AUC$_{\text{TAG}}$	ROC-AUC$_{\text{CLIP}}$	PR-AUC$_{\text{CLIP}}$
8 000	84.78	29.77	90.60	62.94
16 000	85.46	30.42	90.97	64.08
22 050	85.82	30.49	91.25	64.78

Table 6.5: Effect of the sample rate on tag prediction performance.

6.6.4 *Temperature*

The temperature parameter τ in the NT-Xent loss function (Equation 2.3) controls the penalty given to the negative samples. The experiments in Table 6.6 are run using a mini-batch size of 96, which is a smaller mini-batch size than those used in the original SimCLR paper (i.e., $\leq$ 4096 samples per mini-batch) (Chen et al., 2020a). Consequently, the performance difference is marginal. Again, no additional transformations except random cropping are applied in this experiment to isolate the contribution of this temperature parameter.

Temperature	ROC-AUC$_{\text{TAG}}$	PR-AUC$_{\text{TAG}}$	ROC-AUC$_{\text{CLIP}}$	PR-AUC$_{\text{CLIP}}$
0.1	85.82	30.33	91.18	64.10
0.3	85.58	30.35	91.21	64.38
0.5	85.82	30.49	91.25	64.78

Table 6.6: Ablation study of the temperature parameter in the NT-Xent loss function.

Summary

In this chapter, we presented the results of this thesis. We showed that CLMR learns effective representations from raw signals of musical audio. Despite self-supervised pre-training and fine-tuning a linear classifier on the representations, we achieve strong performance on the downstream music classification task. CLMR outperforms the current state-of-the-art waveform-based model on the MagnaTagATune dataset, but it requires longer training to show comparable results on the Million Song Dataset.

We studied the contribution of seven different data augmentations, of which five were studied with altering transformation probabilities, to the robustness of the learned representations. In line with previous research on self-supervised learning, we demonstrated that strong data augmentations during pre-training benefit the downstream task performance. We also showed that too strong augmentations can negatively impact the usability of the representations.

It is common in MIR to have limited labeled data available for training a neural network. Therefore, we tested the efficient classification capability of the CLMR model. We fine-tuned the linear classifier on only 1% of the labels in the dataset, using representations learned during the self-supervised pre-training phase on 100% of the data. The linear classifier still achieved 33.1% PR-AUC, compared to 24.8% PR-AUC with a fully supervised end-to-end model, despite using 100$\times$ fewer labels.

Lastly, we showed that the learned representations are transferable across different musical corpora. When pre-training CLMR on a large, out-of-domain music dataset, and linearly evaluating the representations on another music dataset, the cost is a relatively small performance margin.

7 Interpretability

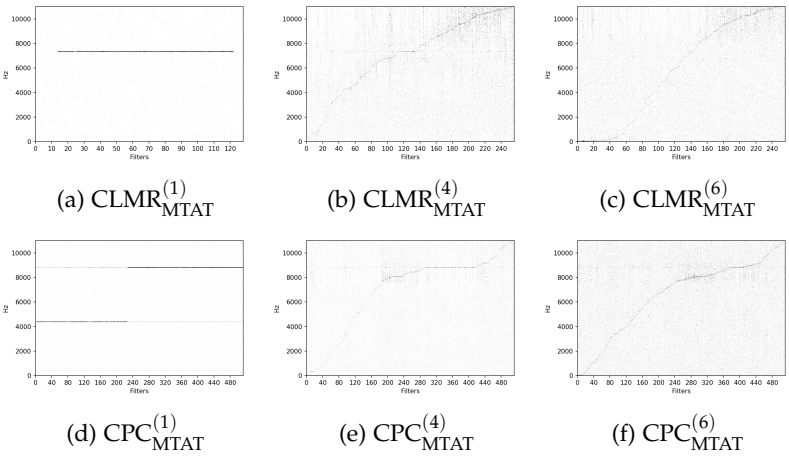
In this chapter, we describe the experiments that were used to qualitatively analyse the features learned by the CLMR model, and get a deeper understanding of the model's learned representations. First, we visualise the filters of the convolutional layers to show what frequencies each filter is most sensitive to. Subsequently, we introduce a factor analysis that guides our qualitative analysis of every dimension in the fine-tuned head. Lastly, we perform a listening experiment to evaluate the (out-of-domain) generalisability of the model.

The following sections describe the experiments that were performed on the frozen, pre-trained feature extractor network [1] in the CLMR framework. In other words, the representations analysed in this chapter are those that were learned in a task-agnostic, self-supervised manner, not those that were learned with the fine-tuned, linear head after the fine-tuning phase.

[1] In our case, the SampleCNN encoder

7.1 Visualising Filters

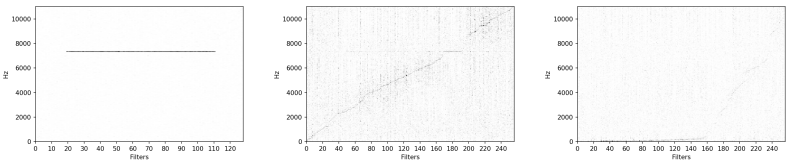
Figure 7.1 shows the magnitude spectrum of the learned filters of the sample-level convolutional layers (layers 1, 4 and 6) for CLMR and CPC, pre-trained on the MagnaTagATune and Billboard dataset. In CLMR, the first layer is sensitive to a single, very small band of frequencies around 7500 Hz, while in higher layers, the filters spread themselves first linearly and then non-linearly across the full range. CPC shows a similar pattern in the lowest layer, but shows a strong activation of two frequencies that span an octave. Interestingly, CLMR pre-trained on the Billboard dataset shows a similar filter structure to fully supervised models that were trained on the MagnaTagATune dataset (Dieleman and Schrauwen, 2014; Lee et al., 2018). For comparison, these are shown in Figure 7.3. The Billboard dataset is significantly less diverse in genre, suggesting the self-supervised model focuses more on such frequency-band related differences than it does for the more genre-diverse MagnaTagATune.



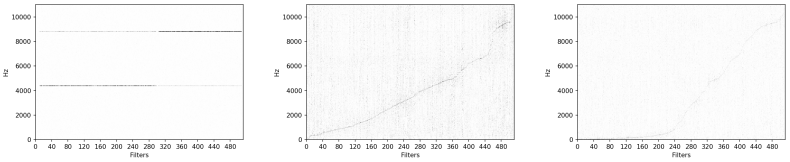
(a) $\text{CLMR}^{(1)}_{\text{MTAT}}$ (b) $\text{CLMR}^{(4)}_{\text{MTAT}}$ (c) $\text{CLMR}^{(6)}_{\text{MTAT}}$

(d) $\text{CPC}^{(1)}_{\text{MTAT}}$ (e) $\text{CPC}^{(4)}_{\text{MTAT}}$ (f) $\text{CPC}^{(6)}_{\text{MTAT}}$

Figure 7.1: Normalised magnitude spectrum of the filters of the self-supervised models in the sample-level convolution layers, sorted by the frequency of the peak magnitude. Gradient ascent is performed on a randomly initialised waveform of 729 samples (close to typical frame size) and its magnitude spectrum is calculated subsequently. Each vertical line in the graph represents the frequency spectrum of a different filter. The first three images are taken from a pre-trained, converged CLMR model, the last three from a CPC model. Both are trained on the MagnaTagATune dataset.



(a) $\text{CLMR}^{(1)}_{\text{Billboard}}$ (b) $\text{CLMR}^{(4)}_{\text{Billboard}}$ (c) $\text{CLMR}^{(6)}_{\text{Billboard}}$



(d) $\text{CPC}^{(1)}_{\text{Billboard}}$ (e) $\text{CPC}^{(4)}_{\text{Billboard}}$ (f) $\text{CPC}^{(6)}_{\text{Billboard}}$

Figure 7.2: Normalised magnitude spectra, generated using the method described in Figure 7.1. These CLMR and CPC models were trained on the Billboard dataset

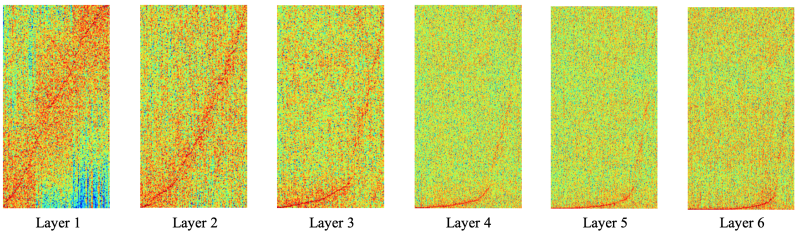


Figure 7.3: Normalised magnitude spectrum of filters from layers 1 - 6 of a fully end-to-end trained supervised SampleCNN network. Figure is taken from Figure 3 in (Lee et al., 2018).

7.2 *Activations*

Figure 7.4 and 7.5 show the mean activations[2] of the last convolutional block of the SampleCNN encoder for every music segment in the test set, for both the MagnaTagATune and Million Song Dataset respectively. In contrast to the MagnaTagATune activations, the last layer of the encoder from the Million Song Dataset activates more broadly for every track, indicating that the encoder responds more flatly. It is important to note that the feature numbers on the x-axis in Figures 7.4 and 7.5 do not correspond to the actual mean activation value - they show how many features were used in the last layer of the SampleCNN encoder. In the following paragraph, we describe a qualitative analysis that uses these activations.

[2] We use the term "activations" to describe the values that are obtained from the last convolutional layer of the SampleCNN encoder, i.e., from the 512 filters, when predicting a segment of musical audio.

7.3 *Listening Experiment*

We have devised an interface which makes it easy to listen to music segments that were run through the CLMR model. For any one of the 512 filters (i.e., features) in the last layer of the pre-trained SampleCNN encoder, we calculate the activations for each music segment. This yields a table of 512 columns of features and X sorted rows based on the activation value of filter. [3]

We take the following, basic approach to qualitatively analyse the features learned by our self-supervised model:

- We calculate the filter activations of the last convolution layer for all music segments in the test set, yielding a 512-dimensional fea-

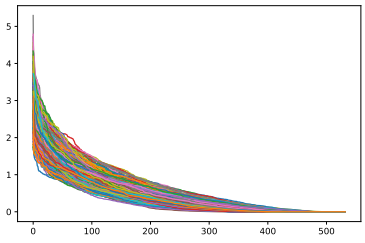


Figure 7.4: Mean activations of 512 features for every music segment, sorted by activation value. Extracted from the encoder of a converged CLMR model trained on the MagnaTagATune dataset.

ture vector for every segment.

- We plot these values in a sortable table and including the corresponding, listenable music segment.
- We choose a single feature from any of the 512 available, and sort the activation values in a descending order.
- We listen to the top-N segments, and write down the sound qualities of those in the upper level (i.e., the first N).

A screenshot of the listening experiment interface is shown in Figure 7.7. We guided our listening experiment with factor analysis first. While this did not yield good results, we later used a manual approach described in paragraph 7.5. We included the factor analysis paragraph for future improvement.

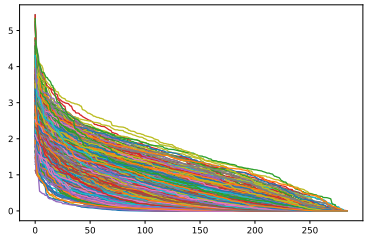


Figure 7.5: Mean activations of 512 features for every music segment, sorted by activation value. Extracted from the encoder of a converged CLMR model trained on the Million Song Dataset.

7.4 *Factor Analysis*

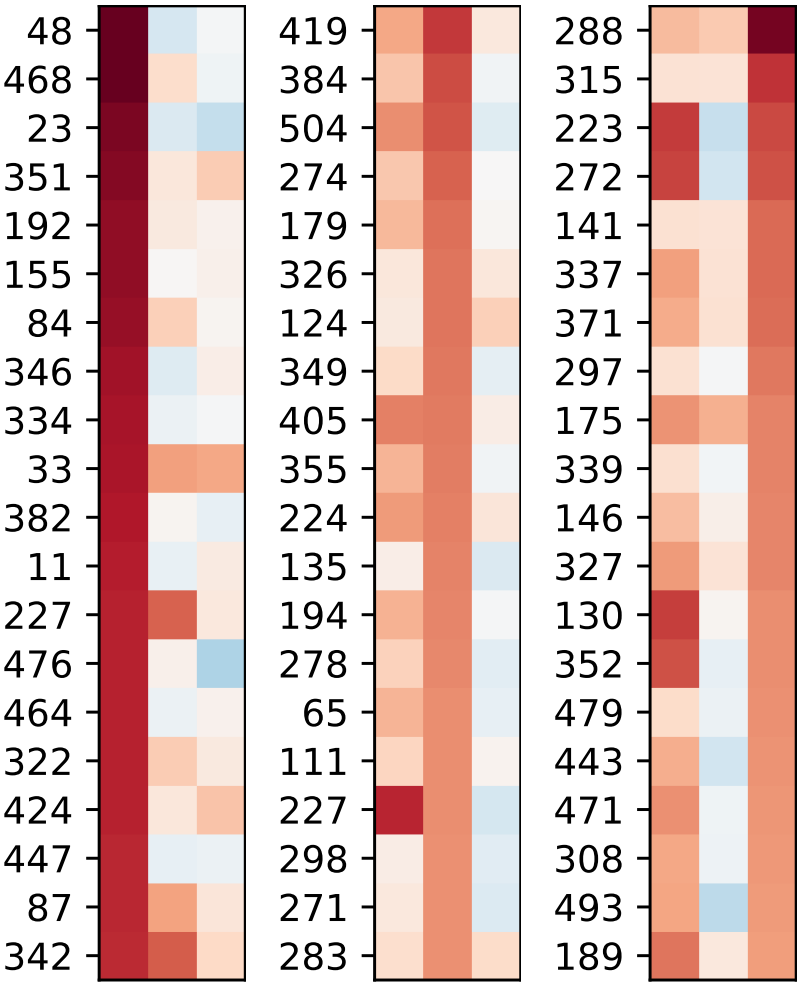
The following analysis is performed using a converged CLMR model that was pre-trained on the MagnaTagATune dataset.

A factor analysis is used to describe correlated variables using a lower number of unobserved variables, i.e., whether variations of many variables reflect those of fewer variables. To guide our listening experiment, we use varimax rotation on the fully connected 512-dimensional layer that is extracted from the linear (or MLP) head after fine-tuning has converged. We choose to use 3 components and subsequently sort each of the 512-dimensional feature vectors by their factor value per component. These are shown in Figure 7.6. This gives us an indication which feature numbers are more closely related.

After listening to segments that highly activate among the relating features obtained from sorting the factors for each component using varimax, we quickly found that it was biased toward grouping loud, techno/(hard-)rock music together for every single component (i.e., for components 1 - 3). To analyse a possible reason for this behavior, we plotted the number of tags in the top-10 activating segments for every feature number in the 512-dimensional feature vector. This is shown in Figure 7.8. It shows the more dominantly present tags that are predicted for every feature number, e.g., 'techno, 'rock', 'beat', 'loud', 'dance'. Most features respond highly to these tags, while other tags like 'vocal' or 'flute' occur less often. These results do not simply translate to the overall tag frequency: the 'guitar' and 'classical' tags occur more in the dataset, but show less activations overall than 'techno' or 'rock'.

While the Factor Analysis grouped together a single, dominant spectral feature, i.e., loudness, and while time is a constraint on the extensivity of this qualitative analysis, we resorted to an easier, but manual intepretation method using *out-of-domain* data.

[3] X being the number of data points in the dataset's test set.



(a) Component 1 (b) Component 2 (c) Component 3

Figure 7.6: Varimax rotation of 3 components on 512 factors (features) on a pre-trained CLMR model on the MagnaTagATune dataset. The figure shows the 20 strongest factors per component. When listening to the corresponding music fragments, we found that loud, techno/(hard-)rock music is present in the strongest factors for every single component.

7.5 *Out-of-domain Generalisation*

To evaluate whether certain features respond to similar spectral qualities, we process both the MagnaTagATune dataset, which the model was pre-trained on, and the Million Song Dataset through the encoder network. In this way, we are able to determine the *out-of-domain generalisation* of the self-supervised model, i.e., is it able to learn features that describe similar sonic qualities of music originating from an entirely different library of music. We extract a 512-dimensional feature vector from the last SampleCNN filters for every segment (as described in Section 7.2), and evaluate whether the same filters respond to the same sonic qualities present in the music segments.

For example, we evaluate whether filter number 412 highly activates on segments of slow-paced jazz music from both the MagnaTagATune and Million Song Dataset, or not. We describe each feature using sound qualities, e.g., in terms of the frequency spectrum, loudness, and ADSR qualities.[4] The filter numbers are pickled by manually listening to the most activating segments of the chosen filter

[4] The attack, decay, sustain, release of the sound.

and, when those musical pieces are sufficiently coherent, we listen to the most activating out-of-domain segments of the same filter.

Filter #51

MagnaTagATune Slow, vocal, string and flute instruments, accompanied by either harpsichord or guitar.

Million Song Dataset Slow, vocal accompanied by guitar. Some segments are up-paced jazz, but those recordings contain a lot of noise.

Filter #61

MagnaTagATune Loud, distorted guitars, vocals, heavy beat, snappy percussion and bass

Million Song Dataset Loud, heavy distorted guitars, hip-hop rapping vocals, percussive bass.

Filter #76

MagnaTagATune Slow string music, vocal, classical, and solo flute, slow attack, not "plucky".

Million Song Dataset Jazz and blues music and some ambient music containing pad sounds (i.e., very slow attack, long decay/sustain/release). The jazz band's brass section plays less "stabby" - but more loose. There is some string music, but overall the sound quality is more slow-moving than fast and pluck-sounding.

Filter #80

MagnaTagATune Very noisy, the full frequency spectrum can be heard, i.e., low bass notes to high notes. You can clearly hear the ambience of the room due to the "breathy" character of the recording.

Million Song Dataset More ambient, you can hear many textures in any register of the frequency spectrum. It is sometimes atonal. The sounds are often slowly moving, and again, the recording is very "breathy".

Filter #105

MagnaTagATune Most recordings contain a plucked or strummed, solo acoustic guitar. Both low- and higher-pitched notes are strummed or plucked in the recordings, and some contain additional instruments like strings and vocals. The guitar is, however, the main instrument. There are two segments of techno music, but both have a very noisy character, i.e., it is a low quality recording.

Million Song Dataset All recordings contain strummed or plucked acoustic guitars, which have a dominant presence in every mix. Most recordings also contain vocals and additional percussive instruments.

There is a single ambient recording among all the guitar-heavy music segments.

Filter #130

MagnaTagATune The recordings contain heavily distored guitars, compressed singing and heavily processed drums. They have a loud character, e.g., rock, metal and heavy metal, and their frequency spectrum is mostly active in the low-mid range.

Million Song Dataset All recordings are either of punk, heavy metal or hard rock music. They are loud and the dominant frequencies are in the low- to mid-range. Again, all instruments are heavily processed in a similar fashion to aforedescribed distortion and compression mixing techniques.

Filter #400

MagnaTagATune All recordings contain one or more vocalists that are dominantly present in the mix. Some recordings contain more reverb than others - but the vocals are always in the foreground of the mix. The singing style of these recordings are opera and non-western, featuring both male and female vocalists.

Million Song Dataset The recordings contain jazz, hip-hop, "old classics" and 60s music. Every recording contains at least one vocalist who is in the foreground of the mix. They are accompanied by a range of backing tracks, e.g., a strong beat to a softly playing string section.

Summary

In this chapter, we analysed the normalised magnitude spectrum of the filters of the self-supervised-based models, CLMR and CPC, and compared them with those that were obtained with fully supervised models. We concluded that the spectrum of the CLMR model that is pre-trained on the Billboard dataset shows a similar pattern to those obtained from fully supervised models. The fully supervised models are prone to fitting to "band-pass"-like filters (Lee et al., 2018). While the Billboard dataset is significantly less diverse in genre, the self-supervised model may focus more on such frequency-band related differences than it does for the more genre-diverse MagnaTagATune dataset.

We also performed an out-of-domain generalisation listening experiment to determine the generalisability of the CLMR model. We found that the pre-trained CLMR model, without fine-tuned head, responds similarly to sound qualities present in music from different datasets, suggesting that it is able to (1) generalise to musical data from an entirely different music library and (2) is able to detect specific timbre, pitch and loudness features.

CLMR Listening Experiment

Encoder

Feature number (0 - 511 hidden layers), comma seperated for multiple: 0,15,30,100,200,432

Finetuned linear classifier

MagnaTagATune Tags
☐ guitar ☐ classical ☐ slow ☐ techno ☐ strings ☐ drums ☐ electronic ☐ rock ☐ fast ☐ piano ☐ ambient ☐ beat ☐ violin ☐ vocal ☐ synth ☐ female ☐ indian ☐ opera ☐ male ☐ singing ☐ vocals ☐ no vocals ☐ harpsichord ☐ loud ☐ quiet ☐ flute ☐ woman ☐ male vocal ☐ no vocal ☐ pop ☐ soft ☐ sitar ☐ solo ☐ man ☐ classic ☐ choir ☐ voice ☐ new age ☐ dance ☐ male voice ☐ female vocal ☐ beats ☐ harp ☐ cello ☐ no voice ☐ weird ☐ country ☐ metal ☐ female voice ☐ choral

Million Song Dataset Tags
☐ rock ☐ pop ☐ alternative ☐ indie ☐ electronic ☐ female vocalists ☐ dance ☐ 00s ☐ alternative rock ☐ jazz ☐ beautiful ☐ metal ☐ chillout ☐ male vocalists ☐ classic rock ☐ soul ☐ indie rock ☐ Mellow ☐ electronica ☐ 80s ☐ folk ☐ 90s ☐ chill ☐ instrumental ☐ punk ☐ oldies ☐ blues ☐ hard rock ☐ ambient ☐ acoustic ☐ experimental ☐ female vocalist ☐ guitar ☐ Hip-Hop ☐ 70s ☐ party ☐ country ☐ easy listening ☐ sexy ☐ catchy ☐ funk ☐ electro ☐ heavy metal ☐ Progressive rock ☐ 60s ☐ rnb ☐ indie pop ☐ sad ☐ House ☐ happy
Submit

Show 10 entries Search:

0	15	30	100	200	432	idx	audio	track_id	clip_id	segment	labels
0.022376619279384613	2.7158913612365723	0.1848115473985672	0.9317127466201782	0.19308944046497345	0	390	0:00 / 0:29		42303	0	classic
0.5030207037925072	2.677877187728882	2.595763683319092	0	0	2.376631498336792	299	0:00		32469	0	techno,dance
0.026872504502534866	2.4450995922088623	0.9709233641624451	0.2756210267543793	0.17556394636631012	0.046880610287189484	442	0:00		48201	0	slow,violin,no vocals,solo
0	2.358271837234497	0.03569496423006058	0	0	0	274	0:00		29956	0	slow,electronic,ambient,quiet
0	2.339538812637329	0.19784429669380188	0.16923396289348602	0.6273652911186218	0	523	0:00		57403	0	
0	2.2877655029296875	0.44966813921928406	0.09701773524284363	0	0	437	0:00		47623	0	slow,choir,choral
0	2.209105968475342	0.29015970230102254	0.15287587046623323	0.28543204069137573	0.022338559851050377	515	0:00		56493	0	slow,vocal,woman

Figure 7.7: Screenshot of the listening experiment interface.

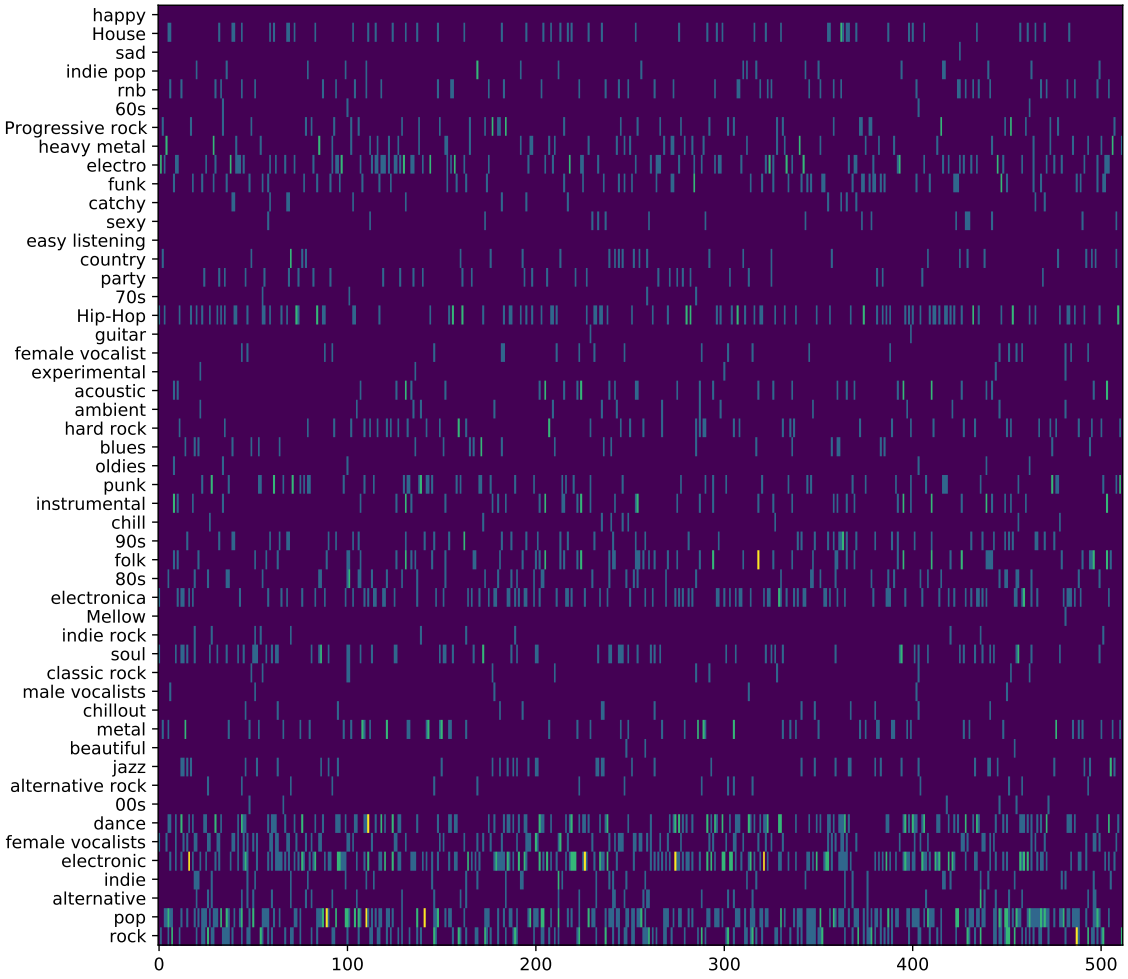


Figure 7.8: Occurrence of tags for the 100 most activating segments for each feature number. Purple indicates low, yellow indicates a high frequency.

8 Conclusion

In this paper, we presented CLMR, a self-supervised contrastive learning framework that learns useful and compact representations of raw waveforms of musical audio.

The framework requires no preprocessing of the input and is trained without ground truth, which enables simple and straightforward pre-training on datasets of unprecedented scale, requiring no human labeling. We tested the learned, task-agnostic representations by fine-tuning a linear classifier on the music classification task on both the MagnaTagATune and Million Song Dataset, achieving state-of-the-art and competitive performance respectively. We showed that CLMR can achieve comparable downstream task performance using 100× fewer labels, and demonstrated the out-of-domain transferability of representations learned from pre-training on entirely different datasets of musical audio. We also demonstrated the out-of-domain generalisability of the CLMR model with a listening experiment.

While we hope this research will advance the field of Music Information Retrieval into a new learning paradigm, we would like to put forward the following recommendations:

- Care must be taken when compiling a dataset of music for pre-training. The biases that are learned inherently in the CLMR model are not investigated in this research. These biases could lead to a potentially skewed prediction performance in favor of a select genre or type of music.
- CLMR requires no human-annotated labels to learn useful representations of music. While normally compensation is provided for human annotation, and, with the absence of this requirement, we recommend instead to compensate artists directly when training this algorithm on their music when developing a commercial application.

To foster reproducible research, we published the source code and pre-trained models of this research on GitHub.[1] The simplicity of training the model without a direct supervised signal and without preprocessing inputs, together with encouraging results obtained with a single linear layer optimised for a challenging downstream task, are exciting developments towards unsupervised learning on raw musical audio.

[1] https://github.com/spijkervet/CLMR

References

Barker, J., Watanabe, S., Vincent, E., and Trmal, J. (2018). The fifth'chime'speech separation and recognition challenge: dataset, task and baselines. *Interspeech*.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. (2011). The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.

Böck, S., Krebs, F., and Widmer, G. (2016). Joint Beat and Downbeat Tracking with Recurrent Neural Networks. In *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR*.

Brewster, D. and Bache, A. D. (1835). *A treatise on optics.* Number 323, [viii, [9]-95 p. Carey, Lea, & Blanchard, Philadelphia.

Burgoyne, J. A., Wild, J., and Fujinaga, I. (2011). An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR*.

Chechik, G., Sharma, V., Shalit, U., and Bengio, S. (2009). Large Scale Online Learning of Image Similarity through Ranking. In Araujo, H., Mendonça, A. M., Pinho, A. J., and Torres, M. I., editors, *Pattern Recognition and Image Analysis*, pages 11–14, Berlin, Heidelberg. Springer Berlin Heidelberg.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020a). A Simple Framework for Contrastive Learning of Visual Representations. *arXiv:2002.05709 [cs, stat]*. arXiv: 2002.05709.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. (2020b). Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*.

Chen, T.-P. and Su, L. (2019). Harmony Transformer: Incorporating Chord Segmentation Into Harmony Recognition. In *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR*.

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 233–240, New York, NY, USA. Association for Computing Machinery.

Defferrard, M., Benzi, K., Vandergheynst, P., and Bresson, X. (2017). Fma: A dataset for music analysis. In *18th International Society for Music Information Retrieval Conference, ISMIR*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Dieleman, S. and Schrauwen, B. (2013). Multiscale approaches to music audio feature learning. In *Proceedings of the 14th International Society for Music Information Retrieval conference*, pages 116–121.

Dieleman, S. and Schrauwen, B. (2014). End-to-end learning for music audio. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968. IEEE.

Doersch, C., Gupta, A., and Efros, A. A. (2015a). Unsupervised Visual Representation Learning by Context Prediction. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1422–1430. IEEE.

Doersch, C., Gupta, A., and Efros, A. A. (2015b). Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision (ICCV)*.

Dosovitskiy, A., Fischer, P., Springenberg, J. T., Riedmiller, M., and Brox, T. (2015). Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1734–1747.

Dosovitskiy, A., Springenberg, J. T., Riedmiller, M., and Brox, T. (2014). Discriminative Unsupervised Feature Learning with Convolutional Neural Networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 766–774. Curran Associates, Inc.

Friston, K. and Kiebel, S. (2009). Predictive coding under the free-energy principle. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521):1211–1221.

Gfeller, B., Frank, C., Roblek, D., Sharifi, M., Tagliasacchi, M., and Velimirović, M. (2020). Pitch Estimation Via Self-Supervision. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3527–3531.

Gidaris, S., Singh, P., and Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. In *Advances in neural information processing systems*, pages 2672–2680.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. (2020). Bootstrap your own latent: A new approach to self-supervised learning. *ArXiv*, abs/2006.07733.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304. PMLR.

Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduc-

tion by learning an invariant mapping. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, page 1735–1742, USA. IEEE Computer Society.

Hamel, P., Lemieux, S., Bengio, Y., and Eck, D. (2011). Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR*, pages 729–734.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

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

Hénaff, O. J., Razavi, A., Doersch, C., Eslami, S. A., and Oord, A. v. d. (2019). Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. (2019). Learning deep representations by mutual information estimation and maximization. *arXiv:1808.06670 [cs, stat]*.

Ioffe, S. and Szegedy, C. (2015a). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Ioffe, S. and Szegedy, C. (2015b). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456.

Kharitonov, E., Rivière, M., Synnaeve, G., Wolf, L., Mazaré, P.-E., Douze, M., and Dupoux, E. (2020). Data augmenting contrastive learning of speech representations in the time domain. *arXiv preprint arXiv:2007.00991*.

Koops, H. V., de Haas, W. B., Burgoyne, J. A., Bransen, J., Kent-Muller, A., and Volk, A. (2019). Annotator subjectivity in harmony annotations of popular music. *Journal of New Music Research*, 48(3):232–252.

Korzeniowski, F. and Widmer, G. (2016). A Fully Convolutional Deep Auditory Model for Musical Chord Recognition. *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6.

Korzeniowski, F. and Widmer, G. (2017). End-to-End Musical Key Estimation Using a Convolutional Neural Network. In *25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece.

Law, E., West, K., Mandel, M. I., Bay, M., and Downie, J. S. (2009). Evaluation of algorithms using games: The case of music tagging. In *Proceedings of the 10th International Society for Music*

Information Retrieval Conference.

Lee, J., Park, J., Kim, K. L., and Nam, J. (2018). SampleCNN: End-to-End Deep Convolutional Neural Networks Using Very Small Filters for Music Classification. *Applied Sciences*, 8(1):150.

Liu, W., Wen, Y., Yu, Z., and Yang, M. (2016). Large-margin softmax loss for convolutional neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 507–516. JMLR.org.

McFee, B., Humphrey, E., and Bello, J. (2015). A software framework for musical data augmentation. In Muller, M. and Wiering, F., editors, *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015*, Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, pages 248–254. International Society for Music Information Retrieval. 16th International Society for Music Information Retrieval Conference, ISMIR 2015 ; Conference date: 26-10-2015 Through 30-10-2015.

Noroozi, M. and Favaro, P. (2016). Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 69–84, Cham. Springer International Publishing.

Oord, A. v. d., Li, Y., and Vinyals, O. (2019). Representation Learning with Contrastive Predictive Coding. *arXiv:1807.03748 [cs, stat]*.

Pascual, S., Ravanelli, M., Serrà, J., Bonafonte, A., and Bengio, Y. (2019). Learning Problem-Agnostic Speech Representations from Multiple Self-Supervised Tasks. In *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pages 161–165.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc.

Pons, J., Nieto, O., Prockup, M., Schmidt, E., Ehmann, A., and Serra, X. (2017). End-to-End Learning for Music Audio Tagging at Scale.

Poole, B., Ozair, S., Van Den Oord, A., Alemi, A., and Tucker, G. (2019). On Variational Bounds of Mutual Information. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5171–5180, Long Beach, California, USA. PMLR.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *4th International Conference on Learn-*

ing Representations, ICLR 2016, San Juan, Puerto Rico, Conference Track Proceedings.

Ravanelli, M., Zhong, J., Pascual, S., Swietojanski, P., Monteiro, J., Trmal, J., and Bengio, Y. (2020). Multi-task self-supervised learning for Robust Speech Recognition. *ArXiv:2001.09239*.

Sermanet, P., Lynch, C., Hsu, J., and Levine, S. (2017). Time-contrastive networks: Self-supervised learning from multi-view observation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 486–487.

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., and Raffel, C. (2020). FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. *arXiv preprint arXiv:2001.07685*.

Sturm, B. L. (2013). The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. *arXiv preprint arXiv:1306.1461*.

Taylor, W. L. (1953). "cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433.

Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S., and Lucic, M. (2020). On Mutual Information Maximization for Representation Learning. In *International Conference on Learning Representations*.

Tzanetakis, G. and Cook, P. (2002). Musical Genre Classification of Audio Signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302.

van den Oord, A., Dieleman, S., and Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in Neural Information Processing Systems 26*, pages 2643–2651.

Wiskott, L. and Sejnowski, T. J. (2002). Slow Feature Analysis: Unsupervised Learning of Invariances. *Neural Computation*, 14(4):715–770.

You, Y., Gitman, I., and Ginsburg, B. (2017). Large batch training of convolutional networks. *arXiv: Computer Vision and Pattern Recognition*.

Zhang, R., Isola, P., and Efros, A. A. (2016a). Colorful Image Colorization. In *European conference on computer vision*, pages 649–666. Springer.

Zhang, R., Isola, P., and Efros, A. A. (2016b). Colorful Image Colorization. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 649–666, Cham. Springer International Publishing.