Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Audio Barlow Twins: Self-Supervised Audio Representation Learning

Author:
Jonah Anton

Supervisors:
Professor Björn Schuller
Harry Coppock

Second Marker:
Dr Pancham Shukla

Submitted in partial fulfillment of the requirements for the MSc degree in Artificial Intelligence of Imperial College London

August 2022

Abstract

This report presents ***Audio Barlow Twins***, a novel self-supervised audio representation learning approach. Audio Barlow Twins adapts the Barlow Twins objective from Computer Vision (CV) to the audio domain. The Barlow Twins objective encourages the empirical cross-correlation matrix between the embeddings of two distorted views of a data sample towards the identity matrix, thereby both enforcing invariance to the applied data augmentations as well as explicitly preventing representational collapse through decorrelation of the individual components of the embedding vectors (*redundancy-reduction*). Audio Barlow Twins utilises a combination of different data augmentations, all of which act directly on the audio data preprocessed as mel-spectrograms, and considers both convolutional and Transformer encoder architectures. We pre-train on the large-scale audio dataset AudioSet, and evaluate the quality of the learned representations on 18 tasks from the HEAR 2021 Challenge, achieving results on a par with the current state-of-the-art for instance discrimination self-supervised approaches to audio representation learning. The impact of the individual components of the learning framework are analysed through extensive ablation studies.

Acknowledgments

I would like to thank my supervisor Professor Björn Schuller for his guidance and advice throughout the duration of this project. I would also like to express my immense gratitude to my co-supervisor Harry Coppock, who generously met with me every week to discuss progress and ideas for the project, even meeting with me whilst he was on holiday. Harry's insight and enthusiasm for this project have been invaluable to me and the success of this project. I'd additionally like to thank my second marker for this project Dr Pancham Shukla, whose expertise at the formative stages of the project were helpful in determining the project's direction. Finally, I'd of course like to thank my parents, Sylvia and Richard, whose loyalty and support has been constant throughout my entire education, and my three brothers, Theo, Sam, and Nathan, for supporting me as always. I'd like to further thank Theo for kindly agreeing to proofread this report.

Contents

Chapter 1

Introduction

Inspired by recent successes in Computer Vision (CV) (e.g. Chen et al. (2020); Grill et al. (2020); Caron et al. (2021)) and Natural Language Processing (NLP) (e.g. Devlin et al. (2018); Liu et al. (2019)) in the generation of universal representations[1] through self-supervised learning methodologies, much recent interest has been dedicated to using self-supervised learning to learn universal representations of audio data (e.g. Baevski et al. (2020); Hsu et al. (2021); Saeed et al. (2020); Niizumi et al. (2021); Gong et al. (2021b); Niizumi et al. (2022b); Huang et al. (2022)). In the context of machine learning (ML), a *universal* representation is one which is able to easily generalise to multiple new contexts (*downstream tasks*) with little to no modification (*fine-tuning*), mirroring the human process of transferring previously learned skills and knowledge to solve new problems. This is evidently a highly desirable property for machines to have, alleviating the current necessity for designing task-specific network architectures and learning frameworks. Within the context of audio, an ideal universal audio representation could be used to solve downstream tasks encompassing multiple different audio domains, such as speech, environmental sound, and music. Self-supervised learning, which constitutes a family of novel unsupervised learning[2] techniques in which the data themselves are used as the supervisory signal for learning, presents an attractive learning paradigm in the pursuit of universal audio representations. Self-supervised methods have been shown to learn more generalisable features than supervised learning (Ericsson et al. (2020)), with far superior data efficiency, since labelled data are not required, which further allows them to leverage large, and abundant, unlabelled datasets.

Many self-supervised learning methodologies have previously been proposed within the audio domain. These include generative[3] approaches, such as SSAST (Gong et al. (2021b)) and Audio-MAE (Huang et al. (2022)). SSAST (Self-Supervised Audio Spectrogram Transformer) leverages a masked spectrogram patch reconstruction objective, using a Transformer (Vaswani et al. (2017) encoder, jointly optimized alongside a discriminative objective where the original unmasked patches are matched to the masked patch embeddings. Audio-MAE adapts the Masked Autoencoder (MAE) (He et al. (2021)) from CV to the audio domain. Inspired by the success of BERT (Devlin et al. (2018)) in NLP, MAE is composed of a Transformer encoder and decoder, where input patches are randomly masked and then reconstructed. Audio-MAE modifies MAE to process audio data preprocessed as spectrograms, masking and reconstructing spectrogram frequency-time patches.

However, whilst such generative approaches have produced state-of-the-art (SOTA) results across many audio tasks, the current SOTA[4] techniques in CV for self-supervised learning (e.g. Zhou et al. (2021); Chen et al. (2021); Assran et al. (2022)) are entirely dominated by instance discrimination based approaches. Instead of learning through reconstruction, instance discrimination approches

[1] A representation is a lower-dimensional and compressed, but highly informative, distillation of an input.

[2] Unsupervised in the sense that no data labels are provided.

[3] Generative self-supervised learning approaches usually learn through occlusion of sections of an input, and then reconstruction or prediction of these occluded sections from the remaining, unoccluded input (e.g. Pathak et al. (2016)).

[4] Using ImageNet-1k (Deng et al. (2009)) top-1 accuracy under linear evaluation with 100% label fraction as a proxy metric for learned representation quality.

train an encoder network to embed similar instances near one another in representation space. In practice, this is typically achieved through generation of two distorted versions, or *views*, of the same input data sample, processing the two views in parallel using a Siamese Network architecture. Barlow Twins (Zbontar et al. (2021)) is one such instance discrimination approach proposed within CV, which encourages the empirical cross-correlation matrix between the embeddings of two views of a mini-batch of data samples towards the identity matrix. The Barlow Twins objective, therefore, through forcing on-diagonal elements of the cross-correlation matrix to one, enforces the instance discrimination principal of embedding similar instances near one another, since embeddings of two distorted views of the same input are encouraged to be perfectly correlated. Moreover, through forcing the off-diagonal elements of the cross-correlation matrix to zero, the Barlow Twins objective minimises the redundancy between the individual components of the extracted embedding vectors, encouraging the learned representations to be maximally informative.

It seems reasonable, therefore, that the Barlow Twins objective, when adapted to the audio domain, would produce robust and generalisable **audio** representations. To this end, in this project we present ***Audio Barlow Twins***, a novel method for the self-supervised learning of audio representations. We consider the application of a variety of different audio distortions, which are applied directly on audio samples preprocessed as (log-spaced) mel-spectrograms, to generate the two views necessary for the Barlow Twins learning framework, as well as several different encoder architectures, including convolutional and Transformer encoders. In fact, the aforementioned SOTA approaches in CV all use a Vision Transformer (ViT) encoder (Dosovitskiy et al. (2020)). The ViT, unlike CNNs, has very weak inductive bias, allowing the structure of the input data to be learned entirely from the data themselves. This, combined with the powerful multi-head self-attention operation, gives the Transformer an incredibly large representational capacity. We anticipate, therefore, that the use of a ViT encoder within the Audio Barlow Twins learning framework may lead to further improvements in learned audio representation quality.

The main contributions of this project are summarised as follows:

1. We present ***Audio Barlow Twins***, an adaptation of the Barlow Twins learning framework from CV to the audio domain. We pre-train on the large-scale audio dataset AudioSet (Gemmeke et al. (2017)), which contains in its full form over 2 million 10 second audio clips covering a total of 527 sound categories. We consider both convolutional and Transformer encoder architectures.

2. We provide a full evaluation of the quality of the representations learned by Audio Barlow Twins on 18 tasks from the HEAR 2021 Challenge (Turian et al. (2022)), which are derived from a total of 15 datasets, and compare to results obtained by other recent SOTA approaches.

3. Extensive ablation studies are performed on the Audio Barlow Twins learning framework, providing insight into the individual contributions of each component. Specific ablation studies are additionally performed for the ViT encoder, with which we further consider partial masking of one of the two spectrogram views, inspired by recent work by Assran et al. (2022).

The remainder of this report is organised as follows. In Section 2 we detail the relevant background for this project, giving the motivation behind Audio Barlow Twins, technical details on the relevant deep learning architectures, and an overview of self-supervised learning. In Section 3 we present the Audio Barlow Twins learning framework. All experimental details are provided in Section 4, including the experimental set-up and hyperparameter tuning (Section 4.1), results and analysis (Section 4.2), as well as ablation studies (Section 4.3) and specific ViT ablation studies (Section 4.4). We conclude in Section 5 and comment on possible future developments. Legal, social, ethical and professional considerations for this project are provided in the Appendix. The codebase developed for this project is available at `https://github.com/jonahanton/MSc_Proj`. This work is being **prepared for submission** to the NeurIPS 2022 Workshop on Self-Supervised Learning, and a first draft of the paper to be submitted is appended to the end of this report.

Chapter 2

Background

2.1 Motivation

2.1.1 The Pursuit of Universal Audio Representations

As discussed in Section 1, self-supervised learning presents two fundamental advantages over supervised learning, the conventional learning paradigm for training deep learning models, in the pursuit of learning universal audio representations. Namely, **1.** feature generalisability, and **2.** scalability.

2.1.1.1 Feature Generalisability

The supervised learning framework requires an annotated dataset D of input, label x, y pairs, $D = \{x_i, y_i\}_{i=1}^{N}$. A predictive model f, parameterised by a set of free variables Θ, is tasked with learning the mapping between the inputs and the labels of D, $\hat{y} = f(x, \Theta) \approx y$. The model parameters Θ are typically optimized by gradient-descent based methods to minimise a loss function $\mathcal{L}$, which measures the discrepancy between the predicted, $\hat{y}$, and true, y, labels,

$$\Theta^{\star} = \arg\min_{\Theta} \sum_{\{x_i, y_i\} \in D} \mathcal{L}(f(x_i, \Theta), y_i). \tag{2.1}$$

Common choices of loss function include the mean squared error (MSE) loss for regression tasks and the cross entropy loss for classification tasks. Supervised learning is therefore an example of a categorical discrimination task, where inputs x that have the same categorical label y are all (indirectly) drawn closer together in feature space, whereas those with different labels are pushed apart[1]. Through enforcing this categorical invariance, Zhao et al. (2020) argue that this minimises the intra-class variation, implicitly assuming that all instances within the same category encode exactly the same semantic content. If there is any misalignment, therefore, between the pre-training task and the downstream task that the pre-trained features are transferred to, information may have been discarded during supervised pre-training that would be useful for this downstream task. This hypothesis is supported by that self-supervised representations, which don't enforce this categorical invariance (by default, since they explicitly use no data annotations), have been shown to transfer consistently significantly better than supervised representations to a broad variety of downstream tasks within the vision domain (Ericsson et al. (2020)). It seems likely, therefore, that this will hold true within the audio domain as well, and that supervised learning can not be used to obtain truly universal representations for audio data. Self-supervised learning offers an attractive alternative.

2.1.1.2 Scalability

Supervised learning requires both the data and their associated labels. The success of supervised learning is therefore reliant on the availability of large, human-annotated datasets, such as ImageNet

[1]The features used in transfer learning from a supervised pre-trained model are usually taken as the model outputs before the final linear classification layer.

(Deng et al. (2009)) for CV and Audioset (Gemmeke et al. (2017)) (Section 3.4.1) for audio, which are expensive and time consuming to acquire. Sun et al. (2017) show that the performance of models trained in a supervised fashion scales approximately logarithmically with the size of the training dataset, highlighting that large labelled datasets are essential for learning in a supervised fashion. As Ericsson et al. (2021) comment, the high cost of data annotations presents a "scalability bottleneck" in the continued development of deep learning systems and AI, particularly as models are increasingly becoming larger and scaled-up as the research community seeks to move towards artificial general intelligence (AGI). Self-supervised learning mitigates this issue directly by learning robust feature representations without the necessity for direct supervision. The self-supervised representations can then be transferred and fine-tuned to a task of interest in a supervised manner with very little (in some cases no) labelled data. This makes self-supervised learning significantly more data efficient than supervised learning. It's also worth noting that the quality of the learned representations from supervised learning relies on the accuracy of the provided annotations. Gong et al. (2021a) observe that the large-scale audio dataset AudioSet (Gemmeke et al. (2017)) contains pervading annotation mistakes (typically false negatives, where an audio clip is missing a number of labels[2]). Once again, by doing away with the requirement for the labels in the first place, self-supervised learning directly mitigates against this issue.

2.1.2 Why Barlow Twins?

2.1.2.1 Intuitive and Explainable Prevention of Representational Collapse

Many state of the art (SOTA) recent self-supervised learning methods build representations based on the core idea of similarity: instances which encode similar semantic content, e.g. two images of a dog, should be embedded near one another in representation space. This concept is known as instance discrimination. There exists a trivial solution, however, which satisfies this requirement, that all representations collapse onto a single constant vector, embedding all instances near (exactly at) all other instances in representation space. These trivial representations encode no meaningful information. This phenomenon is known as representational collapse. Different methods prevent representational collapse in different ways. SimCLR (Chen et al. (2020)) contrasts similar input pairs (which in practice correspond to two distorted versions of the same input) against dissimilar pairs (which are taken by SimCLR as all possible pairs in a mini-batch), pushing together in representation space the former and pushing apart in representation space the latter. Other methods introduce asymmetry into the learning updates (e.g. Grill et al. (2020); Chen and He (2020); Assran et al. (2022); Zhou et al. (2021)), such as through the use of a momentum encoder, which evolves as the exponential moving average (EMA) of the online encoder, stop-gradients and predictor networks. A more detailed overview over self-supervised learning techniques is provided in Section 2.3. Whilst it does intuitively avoid collapse through contrasting against negative samples, SimCLR necessitates the use of extremely large batch sizes in order to obtain a large number of these negative pairs (Chen et al. (2020) use a batch size of 4096), and therefore requires huge computational resources, inaccessible to most research labs. Prevention of collapse through asymmetric learning updates, on the other hand, is highly non-intuitive, sensitive to a number of hyperparameters, and theoretically poorly understood (although some attempts have recently been made, e.g. Tian et al. (2021)). Contrarily, the Barlow Twins (Zbontar et al. (2021)) self-supervised learning method (Section 3.1) intuitively, and by design, prevents representational collapse through decorrelation of the individual components of the extracted representations, making a trivial, constant solution impossible.

2.1.2.2 Directly Enforced Invariance to Audio Distortions

Humans are able to produce abstract and generalisable neurological representations of audio and speech which appear to be insensitive to common distortions of audio signals, such as pitch, frequency, intensity and timbre variations. This allows humans to easily carry out cognitive hearing tasks such as transcription independent of speaker, or speaker identification in a noisy environment. It seems desirable, therefore, to encode similarly these invariances in learned audio representations.

[2]AudioSet (Gemmeke et al. (2017)) is a multi-label dataset.

Whereas methods such as SimCLR do enforce invariances, through pushing embeddings of two distorted views of the same input together in representation space, they do so whilst also enforcing dissimilarity of the embeddings of negative sampled pairs. Further, adaptations of these techniques to audio often sample the negative pairs from time frames from two different audio signals (e.g. Saeed et al. (2020); Fonseca et al. (2020b)). Niizumi et al. (2021) argue that this negative sampling technique is problematic since some sounds may be common between two different signals, such as a chord sequence in music. Barlow Twins, on the other hand, doesn't learn using negative samples, and directly enforces invariances to a set of chosen data augmentations through maximising the correlation (the on-diagonal elements of the cross-correlation matrix) between the embeddings from two distorted versions of an input.

2.1.2.3 Neurological Motivation

As commented by Zbontar et al. (2021), who originally proposed the Barlow Twins self-supervised learning method, the redundancy-reduction principle, as applied in Barlow Twins, originates from the work of neuroscientist David H. Barlow. Barlow (Barlow and Rosenblith (1961)) argues for the redundancy-reduction principle as applied to sensory processing, that sensory relays (a type of neuron) extract signals of "relative high entropy" (high information content) from a "highly redundant" (containing much superfluous information) sensory input. Barlow writes that "sensory relays recode sensory messages so that their redundancy is reduced but comparatively little information is lost." The brain's sensory processing system therefore provides direct motivation and justification for the redundancy-reduction principle of the Barlow Twins self-supervised learning objective.

2.2 Relevant Deep Learning Architectures

2.2.1 Convolutional Neural Networks

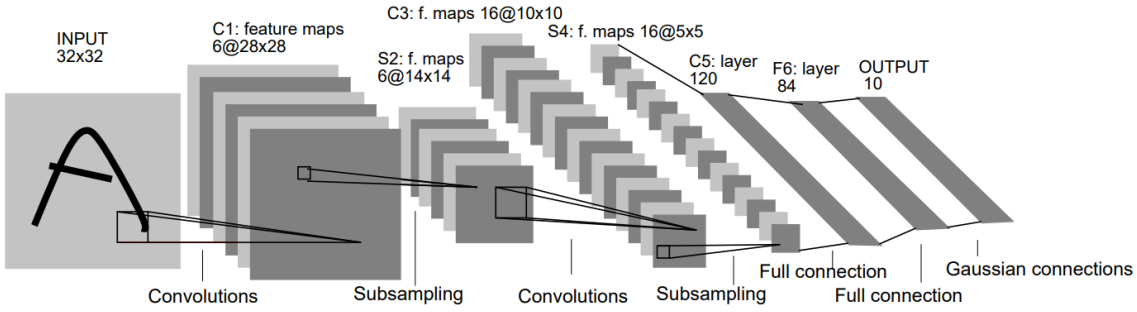


Figure 2.1: **LeNet-5**: The architecture of LeNet-5, one of the first modern convolutional neural networks (CNNs), proposed by Lecun et al. (1998) for handwritten digit recognition. LeNet-5 has a total of 7 layers, interleaving convolutional and subsampling (through averaging pooling) operations, followed by two fully connected output layers. It has a total of $60k$ parameters. Figure taken from Lecun et al. (1998).

Convolutional Neural Networks (CNNs) (Lecun et al. (1998)) are a type of neural network which are designed to process two-dimensional (image) data. Due to their relatively high dimensionality, it is computationally impractical to process images with a feed-forward neural network (FFNN), like is commonly done with one-dimensional vectorized data. Processing a typical image, with a few hundred pixels in either dimension, with only one fully connected layer with one hundred output neurons would already require millions of weights. Further, the processing of a two-dimensional input with a FFNN, where the input is first flattened and treated from then on as an unstructured, one-dimensional input, ignores all topological information available in the image, failing to make use of common image properties such as feature locality and translational equivariance. CNNs, on the other hand, interleave sequential convolutional filters, with shared weights applied across the span of the image, and pooling operations. The convolutional filters, through weight sharing, drastically reduce the number of learnable parameters, and have narrow kernels with small, local receptive fields. In this way, they can extract local features, which is useful when processing two-dimensional data as nearby pixels, whether spatially nearby (image) or temporally nearby (spectrogram), are expected to be highly correlated. Stacking multiple filters together, combined with non-linear activations, allows the model to learn complex, abstract feature maps at each intermediate stage. Pooling operations act as sub-sampling layers, reducing the dimensionality of the input, as well as allowing for aggregation of global features. In recent years, convolutional architectures have led to large performance gains across a multitude of tasks, within both CV (e.g. Krizhevsky et al. (2012); Simonyan and Zisserman (2014)) and audio (e.g. Hershey et al. (2016)).

2.2.1.1 ResNet

He et al. (2015) introduced skip connections, termed residual connections, successfully into convolutional architectures with the ResNet. An ordinary convolutional layer, with input x, produces an output $y = H(x, \Theta)$, with Θ a set of learnable parameters. With a residual connection, the relationship between input and output is transformed to $y = F(x, \Theta) + x$. In this way, we see that the mapping that is to be learned by the model, $F(x, \Theta)$, is the residual from the identity of the original mapping,

$$F(x, \Theta) = H(x, \Theta) - x. \tag{2.2}$$

The model, therefore, is no longer tasked with learning the entire mapping between the inputs and outputs of a given layer, but instead only learning this residual mapping, changing the inductive bias of the model. This allows for the building of deeper architectures, which have a higher capacity, without gradient vanishing.

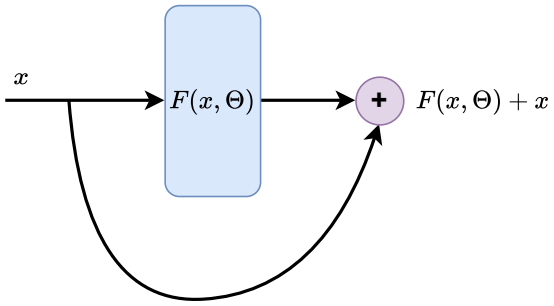


Figure 2.2: **Residual Block**: Schematic depicting a residual block, which are at the core of the ResNet architectures (He et al. (2015)). The output of the residual block is the sum of a non-linear, learnable mapping $F(x, \Theta)$, and a *residual connection*: $H(x, \Theta) = F(x, \Theta) + x$.

2.2.2 The Transformer

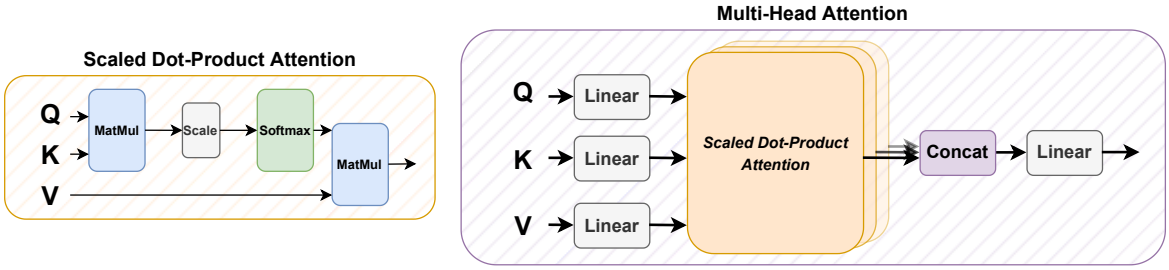


Figure 2.3: **Self-Attention**: The principal component of a Transformer block is the Multi-Head Attention module. Multi-Head Attention (*right*) processes several Scaled Dot-Product Self-Attention (*left*) layers in parallel within different subspaces. Self-Attention allows for the capture of global, long-range dependencies, as each token in the input sequence attends to every other token through query, key-value look-up.

Transformer architectures (Vaswani et al. (2017)) were originally designed for sequence-to-sequence machine translation, and in the past few years a wave of research has seen them produce SOTA results across a broad range of tasks within NLP (e.g. Devlin et al. (2018); Liu et al. (2019); Brown et al. (2020)), CV (e.g. Dosovitskiy et al. (2020); Touvron et al. (2020); Carion et al. (2020)) and audio (e.g. Baevski et al. (2020); Hsu et al. (2021)). Designed to process sequential inputs, Transformers build representations through the use of (self-)attention over the sequence components, learning powerful global dependencies[3]. In direct contrast with convolutional networks, which imposes strong geometric priors (e.g. translational equivariance, feature locality) over the processing of two-dimensional data through the use of pooling operations and small kernels with narrow receptive fields, Transformers preimpose minimal prior knowledge about the structure of the input data, instead allowing the structure of the data to be discovered during training through the self-attention mechanism. The Transformer was originally designed as an encoder-decoder architecture, however in the following we only consider the Transformer encoder, using it in our experiments as a feature extractor.

2.2.2.1 Scaled Dot-Product Attention

As introduced by Vaswani et al. (2017), the main computational mechanism within a Transformer is scaled dot-product (self-)attention, allowing for the capture of long-term global dependencies within a sequence. Khan et al. (2021) explain that a self-attention operation can be seen as an update rule whereby each component of the input sequence is modified dependent on global information aggregated from the entire sequence.

[3]While Transformers can learn global dependencies, they are also clearly capable of learning local ones, whereas convolutional operations can only provide information on the latter.

Consider a set of N queries and M key-value pairs,

$$\{q_i\}_{i=1}^{N}, \{k_i\}_{i=1}^{M}, \{v_i\}_{i=1}^{M}, \tag{2.3}$$

with $q_i \in \mathbb{R}^{d_q}$, $k_i \in \mathbb{R}^{d_q}$, $v_i \in \mathbb{R}^{d_v}$.

The attention mechanism works by first computing, for every query $q_i, i \in [1, N]$, the alignment with each key. Alignment is measured through the scalar product[4]. That is,

$$alignment(i, m) = q_i k_m^T. \tag{2.4}$$

If the query q_i is highly aligned with the key k_m, then the i^{th} component of the output will contain this key's paired value vector v_m with high proportion.

Formally,

$$attention_i = \sum_{m=1}^{M} softmax(q_i k_m^T) v_m, \tag{2.5}$$

where the sum is exteneded over the M key-value pairs and the alignment values have been converted into probabilities through passing a softmax over the distribution[5]. Re-packaging all N queries and M key-value pairs into matrix form, $Q \in \mathbb{R}^{N \times d_q}, K \in \mathbb{R}^{M \times d_q}, V \in \mathbb{R}^{M \times d_v}$, the output of the attention mechanism can be written as

$$attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_q}})V, \tag{2.6}$$

with the output matrix $attention(Q, K, V) \in \mathbb{R}^{N \times d_v}$.

The scaling factor $\sqrt{d_q}$ is introduced into the alignment calculation to prevent a single term dominating the softmax[6]. Scaled dot-product attention has time complexity $O(MNd_q + MNd_v)$. For the Transformer, $M = N$ and $d_q = d_v = d$, and so the time complexity expression becomes $O(N^2 d)$, scaling quadratically with the length of the input sequence.

Considering this mechanism in the context of an input sequence, $X = \{x_i\}_{i=1}^{N}$, $x_i \in \mathbb{R}^d$, the queries, keys, and values are obtained from different learnable projections of the inputs,

$$Q = XW^Q, K = XW^K, V = XW^V, \tag{2.7}$$

where $W^Q \in \mathbb{R}^{d \times d_q}$, $W^K \in \mathbb{R}^{d \times d_k}$, $W^V \in \mathbb{R}^{d \times d_v}$.

2.2.2.2 Multi-Head Attention

Transformers process multiple self-attention blocks in parallel through projecting the inputs into different sub-spaces, performing scaled dot-product self-attention in each subspace. The output of a single multi-head attention block is then the concatenated output of each of the individual heads, which is then projected with a single linear layer to match the dimensionality of the model. The use of multiple attention heads allows a single block to learn many different complex global relationships with minimal additional compute overhead, since the attention mechanisms in each sub-space can be processed in parallel.

Formally,

$$multihead(Q, K, V) = [Z_0, \ldots, Z_h]W^O, \tag{2.8}$$

[4]Hence the restriction that the queries and keys have the same dimensionality, i.e. $d_k = d_q$.

[5]This is known as soft-attention. Hard-attention returns a one-hot vector providing only the value vector for which the corresponding key has the highest alignment.

[6]If q, k are both independent random variables with mean $\mu = 0$, variance $\sigma^2 = 1$, the variance $\sigma^2(qk^T) = \sigma^2(\sum_{i=1}^{d_q} q_i k_i) = \sum_{i=1}^{d_q} \sigma^2(q_i)\sigma^2(k_i) = \sum_{i=1}^{d_q} 1 = d_q$. As a result, division by $\sqrt{d_q}$ gives each term in the softmax unit variance.

where $W^O \in \mathbb{R}^{h \cdot d_v \times d}$, and the output of the i^{th} head is $Z_i = attention(QW_i^Q, KW_i^K, VW_i^V)$, $W_i^Q \in \mathbb{R}^{d_q \times \tilde{d}_q}$, with $\tilde{d}_q$ being the subspace query dimensionality, and similarly for W_i^K, W_i^V.

A full Transformer encoder block combines a multi-head self-attention layer with residual connections, layer normalisations, and point-wise, fully-connected, feed-forward networks.

When used for tasks such as sequence classification, a `[CLS]` (class) token is typically prepended to the input sequence of token embeddings, and the Transformer encoder output for this token after the final Transformer block, z_0, is taken as the global representation for the entire sequence, on which a linear classification head can then be trained.

2.2.2.3 Positional Encodings

The attention operation is permutation invariant. That is, consider the application of a permutation operation P to the queries Q, $Q \leftarrow PQ$. The attention expression then becomes

$$\begin{aligned} attention(Q, K, V) &\leftarrow attention(PQ, K, V) \\ &= softmax(\frac{PQK^T}{\sqrt{d_q}})V \\ &= Psoftmax(\frac{QK^T}{\sqrt{d_q}})V \\ &= Pattention(Q, K, V). \end{aligned}$$

Therefore, the attention mechanism makes no use of the order in which the individual components are passed into the input sequence. To inject positional information into the Transformer, positional encodings are added to the inputs[7].

2.2.2.4 ViT

Dosovitskiy et al. (2020) directly adapted, with as minimal modification as possible, the architecture of the Transformer to process two-dimensional (image) data. Since an image doesn't naturally take the form of a sequence, unlike text (which the Transformer was originally designed to process), the Vision Transformer (ViT) first splits an input image into a series of patches. These patches are then linearly projected and used as the input sequence into the Transformer encoder. In practice, a series of $p \times p$ patches are extracted from an input image through the use of a non-overlapping stride-p $p \times p$ convolution (by default in the original ViT implementation, $p = 16$).

Although many modifications to the original architecture of the ViT have since been proposed, e.g. DeIT (Data efficient Image Transformers (Touvron et al. (2020))), a student-teacher knowledge distillation (KD) based approach with a convolutional (RegNetY (Radosavovic et al. (2020))) teacher, or Swin Transformers (Liu et al. (2021)), introducing a hierarchical structure to the ViT with shifting self-attention windows, the fundamental architectural design (the processing of an image as a series of patches using several sequential multi-head self-attention blocks) has remained unchanged.

2.2.2.5 ViT$_C$

As compared with training CNNs, which have well established training recipes (He et al. (2018)) of optimizers, data augmentations, and standard hyperparameter values, ViTs have been found to be

[7]Typically, positional encodings are left as free parameters and learned during training. Another common choice for positional encodings, however, is using fixed sinusoidal embeddings. That is, for the i^{th} component of the n^{th} element in the sequence, the positional encoding (PE) is

$$\text{PE}(n, 2i) = sin(\frac{n}{10000^{2i/d_q}}) \tag{2.9}$$

$$\text{PE}(n, 2i+1) = cos(\frac{n}{10000^{2i/d_q}}) \tag{2.10}$$

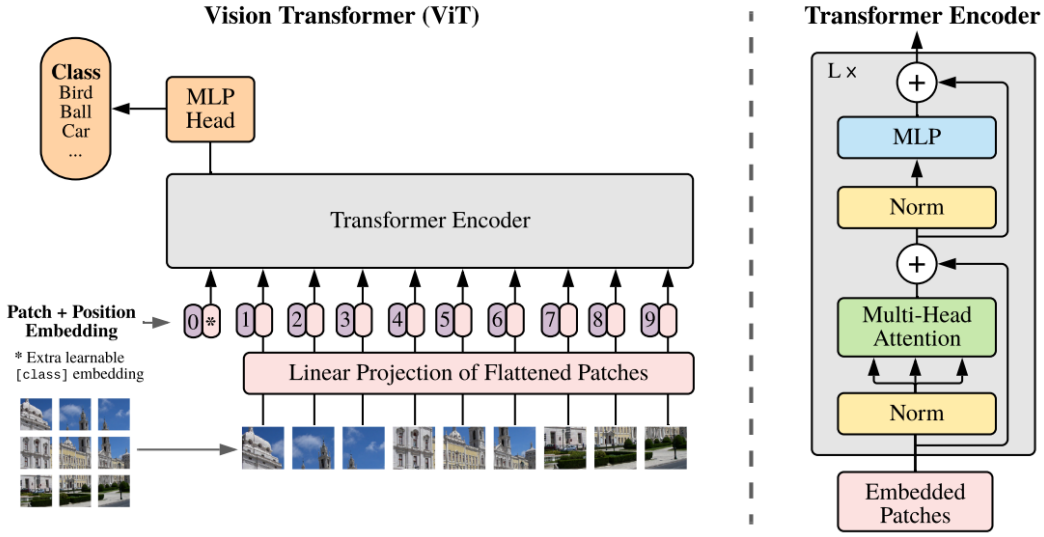
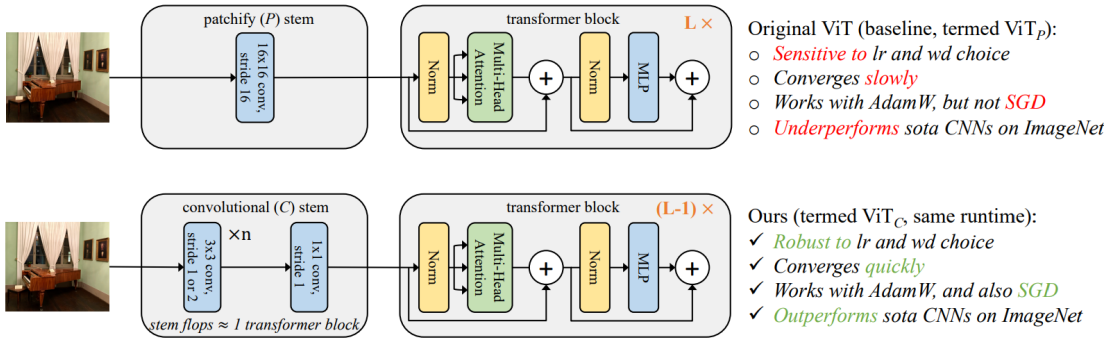
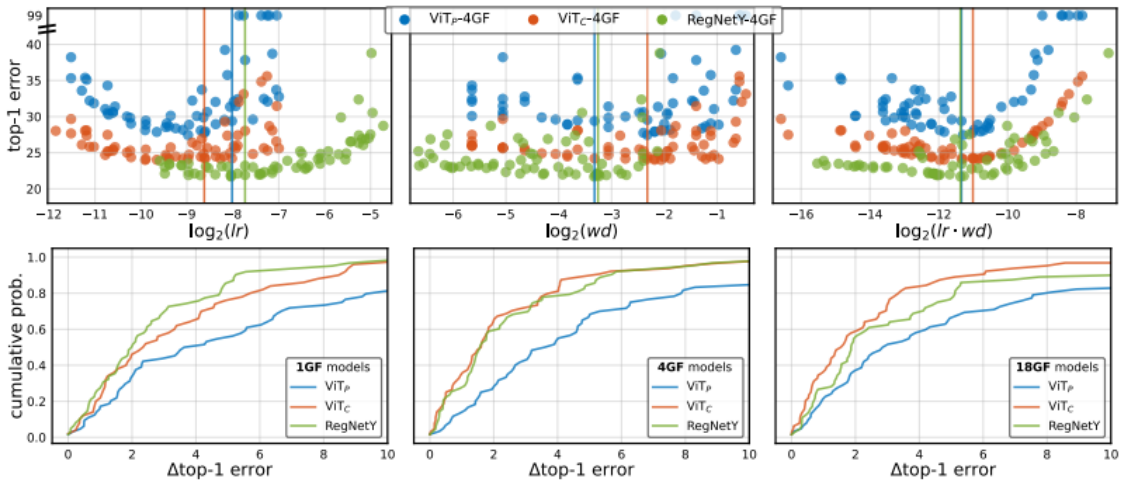


Figure 2.4: **Vision Transformer**: The Vision Transformer (ViT) (Dosovitskiy et al. (2020)) architecture minimally adapts the standard Transformer encoder (*right*) for the processing of image data. Image patches are first extracted using a single stride-p $p \times p$ convolution, producing a series of $p \times p$ image patches which are used as input tokens into the Transformer encoder. By default, $p = 16$. Figure taken from Dosovitskiy et al. (2020).

much more difficult to optimize (Chen et al. (2021); Xiao et al. (2021)). ViTs show high sensitivity to the choice of optimizer (Touvron et al. (2020)) (typically trained with AdamW (Loshchilov and Hutter (2017)) and show high learning instabilities with stochastic gradient descent (SGD)), as well as optimizer hyperparameters, such as learning rate (*lr*) and weight decay (*wd*). Chen et al. (2021) propose a small trick to improve ViT learning stability, where they freeze the stride-p $p \times p$ patch projection layer to its random initialization in the Transformer stem during training. This small adjustment stabilises training and benefits model performance (ImageNet-1k top-1 accuracy increases by over 1%). Xiao et al. (2021) similarly hypothesise that the instabilities associated with training ViTs arise from the large stride-p $p \times p$ patch projection, which they term the *patchify stem*. They propose to replace the patchify stem with a *convolutional stem*, where the single stride-p $p \times p$ convolution is replaced by a stack of ~ 5 3×3 convolutions with strides 1 or 2, followed by a final 1×1 convolution to match the input dimensionality of the Transformer encoder. Xiao et al. (2021) refer to the modified ViT with a convolutional stem as a ***ViT***$_C$. The ViT$_C$ trains faster than the original ViT, shows stable training with both optimizers AdamW and SGD, is less sensitive to the choice of *lr* and *wd*, and enchances model performance, increasing ImageNet-1k top-1 accuracy by $\sim 1 - 2\%$.



(a)



(b)

Figure 2.5: **Vision Transformer with a convolutional stem**: Xiao et al. (2021) find that replacing the initial stride-p $p \times p$ patch projection, which they term the *patchify stem*, in the ViT with a stack of ~ 5 3×3 convolutions, followed by a final 1×1 convolution to match the dimensionality d of the Transformer encoder, leads to much more stable training and improved model performance. Xiao et al. (2021) coin these series of convolutions the *convolutional* stem, as shown in **a)**, and the resulting modified ViT is termed the ViT$_C$. The improved stability of the ViT$_C$ is demonstrated in **b)**. The upper plots show the variation of ImageNet-1k top-1 error with optimizer (AdamW) *lr/wd*. The lowest plots show error distribution functions (EDFs), where an EDF shows, for a given error, the proportion of results with value lower than this error. From the EDFs, the ViT$_C$ (across model sizes: 1GF $\approx$ ViT-*T*, 4GF $\approx$ ViT-*S*, 16GF $\approx$ ViT-*B*) shows similar stability to the RegNetY and far superior stability to the ViT, with a much higher fraction of *lr/wd* pairs being within a small Δtop-1 error of peak performance. All figures taken from Xiao et al. (2021).

2.3 Self-Supervised Learning

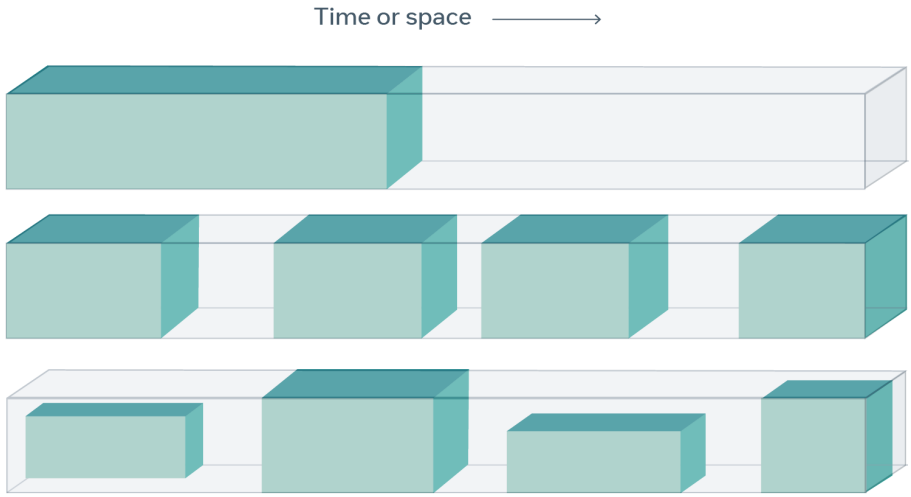


Figure 2.6: **Self-Supervised Learning is predictive learning**: The general self-supervised learning paradigm is summarised by Yann Lecun (Lecun and Misra (2021)) as a form of predictive learning, where the model is tasked with predicting the occluded parts of the input (in grey) from the remaining, unoccluded, parts of the input (in green). Figure taken from Lecun and Misra (2021).

Self-supervised learning refers to the learning paradigm where feature representations are learned through using the data themselves as the "supervisory signal" (Lecun and Misra (2021)) for learning.

Yann Lecun summarises self-supervised learning as "filling in the blanks" (Lecun and Misra (2021)), where a part of the input, whether that be an audio clip, an image, a video, or text, is occluded and the model must predict these missing sections from the remaining input. Direct examples of this technique include image inpainting (Pathak et al. (2016)), where the objective is to reconstruct masked sections of input images, and masked audio frame prediction (e.g. Liu et al. (2020)), which similarly requires reconstruction of masked time frames of input audio signals. Self-supervised learning, therefore, is effectively a form of predictive learning, using the relationships between the different parts of data themselves, and not using any data labels, to build robust feature representations.

After the building of these representations, which is a process known as the pretext task[8], the general self-supervised learning framework then continues through transferring these learned representations to a downstream task, initialising the model with the pre-trained weights. A small multi-layer perceptron (MLP), often consisting of only one or two linear layers, is attached to the pre-trained model, and the model is trained in a supervised manner on the target dataset. Linear evaluation refers to the scenario in which the weights of the pre-trained model are frozen and only the MLP is trained, whereas in end-to-end fine-tuning the entire model is re-trained alongside the MLP. As formalised by Ericsson et al. (2021), if the self-supervised pre-trained feature extractor is parameterised by weights θ, $h_\theta(\cdot)$, the MLP classification head by weights ϕ, $g_\phi(\cdot)$, and the downstream target dataset D consists of N data label, pairs, $D = \{x_i, y_i\}_{i=1}^N$, the linear evaluation objective is

$$\phi^\star = \arg\min_\phi \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(g_\phi(h_\theta(x_i)), y_i), \tag{2.11}$$

[8]Pretext in the sense that it's not the actual task of interest, but rather pre-training to acquire useful knowledge that is, ideally, transferable to a downstream task.

whereas the fine-tuning objective is

$$\theta^\star, \phi^\star = \arg\min_{\theta,\phi} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(g_\phi(h_\theta(x_i)), y_i), \tag{2.12}$$

with, for both linear evaluation and fine-tuning, the weights θ are initialised with those learned from self-supervised pre-training.

2.3.1 Instance Discrimination

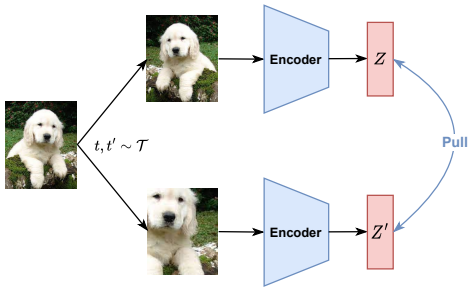


Figure 2.7: **Instance Discrimination**: Instance discrimination approaches to self-supervised learning build a meaningful representation space through encoding similar images near one another. In this diagram, two distorted versions, *views*, of an image of a dog are processed in parallel by two encoder networks, which constitute the two 'arms' of a Siamese Network. The instance discrimination objective then pulls together their representations.

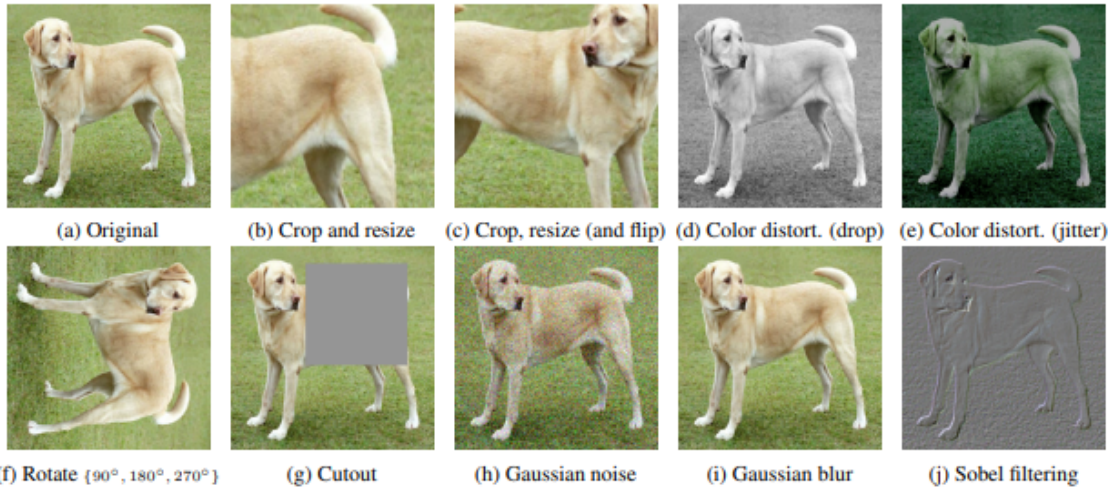


Figure 2.8: **Image Distortions**: Common image distortions applied to generate image views. Notice how the semantic content of the image (a dog) remains invariant to the applied augmentations. Figure taken from Chen et al. (2020).

Many SOTA self-supervised learning frameworks follow an instance discrimination approach, using Siamese networks. A brief overview and a few relevant SOTA instance-discrimination self-supervised learning methods will now be discussed[9].

Differing from the "filling in the blank" approach as discussed by Yann Lecun, instance discrimination approaches are built on the core idea of similarity: instances which encode similar semantic content

[9]There exist many other types of self-supervised learning frameworks, such as generative and predictive approaches, which, whilst highly relevant to the field at a large, do not directly relate to or inspire our experiments. Nonetheless, detailed information can be found from many excellent surveys which provide a broader overview, such as Ericsson et al., 2021.

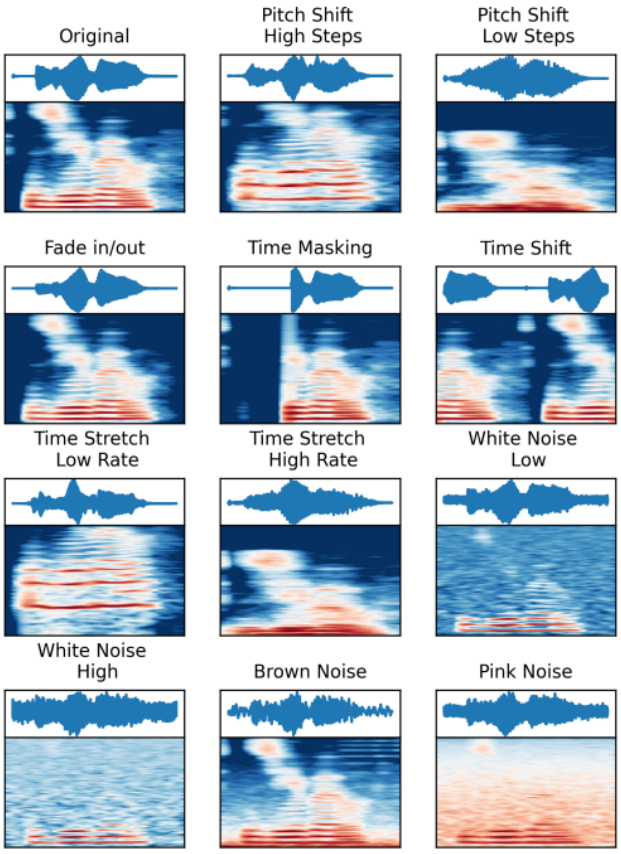


Figure 2.9: **Audio Distortions**: Common audio distortions applied to raw audio waveforms to generate different distorted version of an input clip (*views*). Figure taken from Al-Tahan and Mohsenzadeh (2020).

should be embedded near one another in representation space. In this way, a meaningful latent space is constructed. These methods make use of a Siamese network, where each 'arm' of the network processes a different view of the data sample. The extracted feature representations of the two views are then pushed together.

The two views are commonly generated[10] through stochastic application of distortions (often referred to as *data augmentations*) to the input data sample. Through using two distorted views of a data instance as a proxy for a positive pair of different instances within a dataset, it is trivially guaranteed that the two samples processed by the different arms of the Siamese network each contain the exact same semantic content. In CV, image distortions include Random Resize Crop (RRC) (random crop followed by resizing back to the original image size), horizontal flip, Gaussian blur, solarization, and colour jitter. Audio distortions can either act directly on raw audio waveforms (examples include pitch shift, time shift, time stretch, time masking, and fade in and out) or on the extracted mel-spectrogram (Section 2.4.1), which can be treated similarly to an image (with some minor modifications) and so common CV image distortions can be borrowed. A more in depth discussion on how the data views are generated in our experiments are provided in Section 3.2.

Solely enforcing representational similarity of positive pairs is vulnerable to mode collapse onto a constant vector for all inputs, a phenomenon known as representational collapse. Different self-

[10]There exist methods which generate the two views in different ways, not through the application of data augmentations (e.g. Al-Tahan and Mohsenzadeh (2020)).

supervised instance discrimination methods implement different strategies to mitigate against this.

2.3.1.1 Contrastive Instance Discrimination

Contrastive instance discrimination approaches prevent representational collapse through the use of negative pairs (encoding different semantic content, e.g. an image of a dog and an image of a cat), which are forced apart in representation space. An established contrastive instance discrimination self-supervised learning method, from CV, is **SimCLR** (A Simple Framework for Contrastive Learning of Visual Representations) Chen et al. (2020). SimCLR consists of four stages in its learning framework.

1. Generation of views: Two views of an image x, $\tilde{x}_i = t(x)$ and $\tilde{x}_j = t'(x)$, are generated from two sets of data augmentations randomly sampled from the data augmentation module, $t, t' \sim \mathcal{T}$. The data augmentation module for SimCLR consists of RRC, colour jitter, and Gaussian blur.

2. Feature extraction: The image views $\tilde{x}_i, \tilde{x}_j$ are processed by the encoder network $h_\theta(\cdot)$ to obtain their feature representations $r_i = h_\theta(\tilde{x}_i), r_j = h_\theta(\tilde{x}_j)$. The encoder network $h_\theta(\cdot)$ is a ResNet (He et al. (2015)).

3. Projection head: A small multi-layer perceptron (MLP) with one hidden layer, $g_\phi(\cdot)$, is applied to the features: $\tilde{z}_i = g_\phi(r_i), \tilde{z}_j = g_\phi(r_j)$.

4. Contrastive loss: Consider a mini-batch $\{x_k\}_{k=1}^N$. Given the augmented set $\{\tilde{x}_k\}_{k=1}^{2N}$, the contrastive loss aims to correctly identify the corresponding view $\tilde{x}_j$ for a given $\tilde{x}_i$ in the set $\{\tilde{x}_k\}_{k\neq i}$. Formally,

$$\ell(\tilde{x}_i, \tilde{x}_j) = -\log \frac{\exp(s(z_i, z_j)/\tau)}{\sum_{k=1,\neq i}^{k=2N} \exp(s(z_i, z_k)/\tau)}, \tag{2.13}$$

   where $s(a, b)$ is the cosine similarity between a and b, and τ is a temperature parameter. This loss is known as NT-Xent (Normalised Temperature-scaled cross(X) entropy loss), and is also referred to in the literature as InfoNCE. The loss is then symmetrized as

$$\ell_{\text{total}}(\{\tilde{x}_i, \tilde{x}_j\}) = \ell(\tilde{x}_i, \tilde{x}_j) + \ell(\tilde{x}_j, \tilde{x}_i), \tag{2.14}$$

   and averaged over all positive pairs in the mini-batch. Observe that SimCLR does not explicitly sample negative pairs, instead considering all pairs of image views which do not originate from the same base image as negative examples.

2.3.1.2 Non-contrastive Instance Discrimination

As opposed to sampling negative pairs, non-contrastive instance discrimination approaches usually introduce asymmetry into the learning framework to prevent representational collapse[11].

Consider, for example, **BYOL** (Bootstrap Your Own Latent) (Grill et al. (2020)), named as such since one network "iteratively bootstraps" (Grill et al. (2020)) the outputs of another as a target signal for representation learning. In other words, a sequence in increasing representation quality is produced, where at each iterative step the quality of one encoder's representations is improved using the other encoder's representations as a target. The two networks, which constitute the two arms of a Siamese network, are trained together, and termed the online encoder and the target encoder (the online encoder bootstraps the output of the target encoder). The online encoder is parameterised by a set of weights θ, and the target encoder, which has identical architecture to the online encoder, is parameterised by a different set of weights ξ.

Similar to with SimCLR, two augmented views[12] v, v' of an image x are generated through applying two randomly sampled sets of data augmentations, $t, t' \sim \mathcal{T}$, $v = t(x), v' = t'(x)$. The online

[11] There exists other non-contrastive instance discrimination approaches which do not use asymmetric learning updates, such as clustering e.g. SwaV (Caron et al. (2020)) and DeepCluster (Caron et al. (2018)), although these are not discussed directly in this report (the SwaV objective is briefly mentioned in passing in Section 2.3.1.3 as it is used by Assran et al. (2022)).

[12] We choose to use the same notation as the original BYOL paper (Grill et al. (2020)).

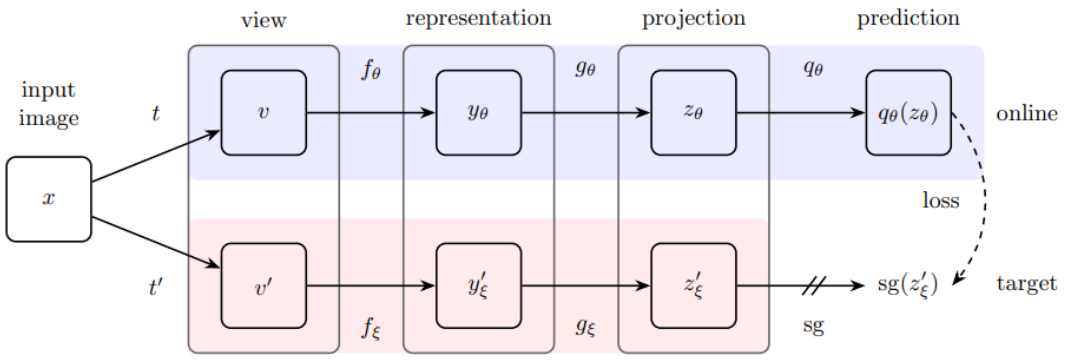


Figure 2.10: **BYOL**: The learning framework for BYOL (Bootstrap Your Own Latent) (Grill et al. (2020)), consisting of an online encoder and a target encoder. The online encoder "iteratively bootstraps" (Grill et al. (2020)) the outputs of the target encoder, producing during training a sequence of representations of increasing quality. The weights of the target encoder, ξ, evolve as the exponential moving average (EMA) of those of the online encoder, θ. Figure taken from Grill et al. (2020).

network outputs a representation y_θ and a projection z_θ, and the target network outputs y'_ξ and a target projection z'_ξ. The online network generates a prediction $q_\theta(z_\theta)$ of z'_ξ, using an additional *predictor* network (a small MLP), $q_\theta(\cdot)$. The loss is computed as the mean squared error between the ℓ_2-normalised predictions $\bar{q}_\theta(z_\theta)$ and target projections $\bar{z}'_\xi$,

$$\mathcal{L}_{\theta,\xi} = ||\bar{q}_\theta(z_\theta) - \bar{z}'_\xi||^2. \tag{2.15}$$

The loss is symmetrized by feeding v to the target network and v' to the online network,

$$\mathcal{L}^{\texttt{BYOL}}_{\theta,\xi} = \mathcal{L}_{\theta,\xi}(v, v') + \mathcal{L}_{\theta,\xi}(v', v). \tag{2.16}$$

$\mathcal{L}^{\texttt{BYOL}}_{\theta,\xi}$ is only minimised with respect to θ[13], and the parameters of the target network ξ are set as the slowly moving exponential average of θ,

$$\xi \leftarrow \tau\xi + (1-\tau)\theta. \tag{2.17}$$

Despite the BYOL learning framework allowing trivial solutions (representational collapse), Grill et al. hypothesise that such constant solutions are dynamically unstable due to the asymmetric structure of the learning updates[14] (EMA + stop gradient + predictor network).

2.3.1.3 Transformer Based Approaches

As Dosovitskiy et al. (2020) found, ViTs (Dosovitskiy et al. (2020)) benefit significantly from longer training schedules and larger datasets, since the multi-head self-attention operation integral to the Transformer, which processes information globally, has a much greater representational capacity than convolutions, but lacks (by design) convolutions' strong geometric inductive biases (feature locality and equivariance) and so performs comparably worse when trained with smaller datasets, for less time. Since ViTs are so data-hungry, they naturally lend themselves to large-scale self-supervised pre-training, which allows leveraging of large, unlabelled datasets.

DINO (self-distillation with no labels) (Caron et al. (2021)) follows a very similar learning framework to BYOL (Grill et al. (2020)), using an online encoder (termed the *student* network, $g_{\theta_s}(\cdot)$) and a target encoder (termed the *teacher* network, $g_{\theta_t}(\cdot)$), whose weights are not dynamically learned

[13]This is implemented in practice through the use of a stop gradient on the outputs of the target encoder. In `PyTorch`, this can be easily done using the `torch.no_grad()` context manager, which disables gradient calculation for all operations performed within its scope.

[14]Tian et al. (2021) provide more theoretical insight to why BYOL does not suffer from representational collapse.

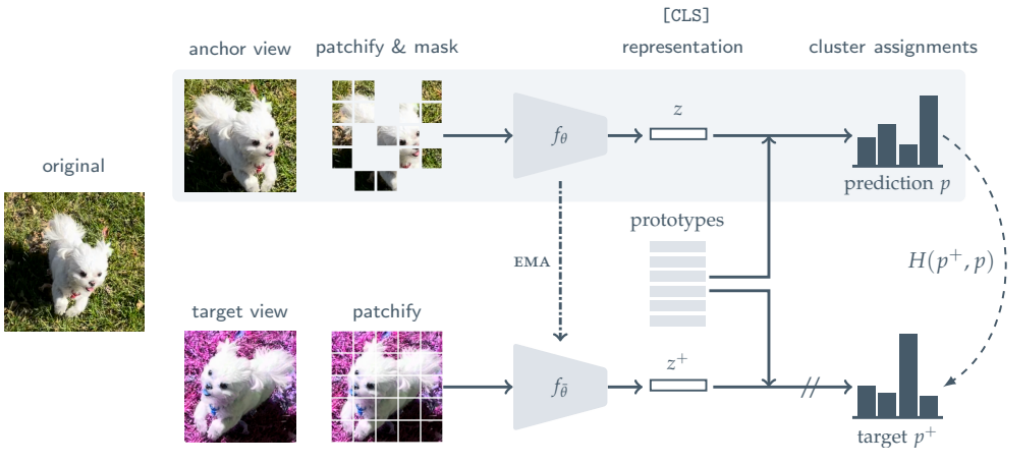


Figure 2.11: **MSN**: The learning framework for MSN (Masked Siamese Networks) (Assran et al. (2022)). MSN introduces a novel masking strategy, randomly dropping a subset of the image patches of the anchor view (as shown in the upper arm of the Siamese network) before input into the ViT encoder. In this way, MSN performs "implicit denoising" (Assran et al. (2022)) at the representation level, matching the representations of the masked and unmasked image views. Figure taken from Assran et al. (2022).

and instead updated as the EMA of the online encoders'. However, crucially, DINO uses a ViT encoder, instead of a convolutional one, taking the output of the prepended `[CLS]` token as the overall image representation that is passed to the projection head, whose outputs the self-supervised loss is calculated on. Further, unlike BYOL, DINO does not use a predictor network (it's found to degrade performance) nor a MSE loss, but instead uses a cross-entropy loss between the student and teacher outputs (after a temperature-scaled softmax operation) in order to try to match their probability distributions.

That is, the objective for the weights θ_s of the student network is

$$\theta_s^\star = \arg\min_{\theta_s} H(P_t(x), P_s(x)), \tag{2.18}$$

where $P_s(x) = softmax(g_{\theta_s}(x), \tau_s)$, and similarly for $P_t(x)$ (with the addition of a centering operation on the teacher outputs, see the original publication (Caron et al. (2021)) for more details), and $H(a, b) = -a \log b$.

DINO further utilises the *multi-crop* data augmentation strategy, first proposed by Caron et al. (2020). This is where, instead of generating only two image views, which are still generated as usual and termed the *global* views x_1^g, x_2^g, in addition a set of V *local* image views, of smaller resolution covering a smaller area of the original image, are generated, $x' \in V$. The objective is then computed over all possible "local-to-global" (Caron et al. (2021)) correspondences,

$$\theta_s^\star = \arg\min_{\theta_s} \sum_{x \in \{x_1^g, x_2^g\}} \sum_{x' \in V, x \neq x'} H(P_t(x), P_s(x')), \tag{2.19}$$

where all views (both local and global) are passed through the student network and only the two global views are passed through the teacher network.

Similar to DINO, **MSN** (Masked Siamese Networks, Assran et al. (2022)) uses a ViT encoder and a student-teacher Siamese network learning framework, with the teacher weights updated as the EMA of the students'. However, aside from the way in which the probability distributions $P_s(x)$ and $P_t(x)$ are calculated[15], MSN introduces a random masking strategy to the image view(s) processed by the

[15] Recall, in DINO (Caron et al. (2021)) the student and teacher probability distributions are calculated using a temperature-

student (referred to as the anchor views). This is where a random subset of the image patches of each anchor view is dropped (but leaving the target view unchanged) and discarded before input into the student encoder. The use of a ViT encoder allows these masked patches to be completely disregarded and not processed in the extraction of the output representations, since they can be simply not passed into the ViT[16].

Since the MSN objective matches the representations of the student and teacher views, the network is tasked with performing "implicit denoising" (Assran et al. (2022)) at the representation level, in contrast with approaches such as the Masked Autoencoder (MAE) (He et al. (2021)) which perform direct denoising of masked patches at the pixel level through a masked-patch reconstruction objective. Assran et al. (2022) also note that through masking patches, the computational and memory requirements of MSN are significantly reduced. For example, randomly masking 70% of the input patches for the anchor views is reported to halve training computation and memory, allowing MSN to more easily scale to larger ViT architectures (which have higher representational capacities). The representations produced by MSN outperform DINO and all other self-supervised learning methodologies in CV for low-shot[17] ImageNet evaluation[18], as well as outperforming DINO and on-par with the SOTA for ImageNet-1k linear evaluation and end-to-end fine-tuning[19].

scaled softmax over the network outputs (with an additional centering operation for the teacher outputs). In MSN (Assran et al. (2022)), the probability distributions are instead calculated as a soft code assignment of the representations to a set of K learnable prototypes, similar to as in SwAV. MSN enforces a secondary loss in the form of a mean entropy maximisation (`ME-MAX`) regularizer, which encourages full use of the set of prototypes, thereby preventing representational collapse. Full details can be found in the original publication (Assran et al. (2022)).

[16]Unlike e.g. BERT (Devlin et al. (2018)), MSN does not replace the masked patches with a learnable `[MASK]` token.

[17]Low-shot evaluation refers to training on a downstream task using a small fraction of the available training labels. In the case of MSN (Assran et al. (2022)), this corresponds to using 1, 2, 5, and $\sim$ 13 (1% of all labels) images per class).

[18]Results found in Tables 1, 2 (pgs 7, 8) in the original MSN publication (Assran et al. (2022)).

[19]Results found in Tables 3, 4 (pg 9) in the original MSN publication (Assran et al. (2022)).

2.4 Audio Representation Learning

2.4.1 Input Representations

In its most low-level form, an audio signal can be represented as a raw waveform. The raw waveform is encoded from the original signal using pulse code modulation (PCM) at a specific sampling frequency f_s. From the Nyquist-Shannon theorem, the highest frequency that can be reproduced from the encoded signal is half that of the sampling frequency, $f_{nq} = \frac{1}{2}f$, and so when using, for example, a sampling frequency of 16 kHz (as we do in all our experiments), only frequencies below the Nyquist frequency of 8 kHz should be considered. The raw waveform consists of a vector of scalar amplitudes sampled at different points in time. Whilst raw waveforms are a rich source of information about the original signal, they require high computation and memory to process[20], since, for example, a one second signal sampled at 16 kHz generates a waveform with dimensionality 16,000. Raw waveforms, therefore, are often first converted to spectrograms, which are a time-frequency representation of audio signals.

2.4.1.1 The Mel-Spectrogram

A spectrogram can be obtained from applying the Short Time Fourier Transform (STFT) to a series of small, overlapping windows of a raw waveform.

Consider a raw waveform $x \in \mathbb{R}^{t*f_s}$, where t is the signal length (seconds) and f_s is the sampling frequency (Hz). Using a sliding window of length l, with step s, a series of T frames $X = \{X_i\}_{i=1}^{T}$ can be extracted, where $X_i \in \mathbb{R}^N, N = l * f_s$. The Discrete Fourier Transform (DFT) is then applied to each frame to obtain

$$\tilde{X}_i = \sum_{n=1}^{N} X_i(n) e^{-j2\pi kn/N} \tag{2.20}$$

where $k = 1, \ldots, K$ are the frequency coefficients.

The spectrogram values are taken as the absolute magnitudes of the STFT, discarding all phase information. The (log-scaled) mel-spectrogram can be obtained through logs and converting to the mel scale,

$$\text{mel}(f) = 2595 * \log_{10}(1 + f/700) \tag{2.21}$$

The mel-spectrogram is often used over the original spectrogram as it has been shown that the mel frequency spacing more closely resembles the distance between frequencies as perceived by the human ear.

2.4.2 Audio Self-Supervised Learning

Many self-supervised learning methods have been proposed to learn generalisable audio representations. We provide an overview[21] of a few methods that build directly on the instance discrimination methods developed in CV as discussed in Section 2.3.1.

2.4.2.1 (Audio) Contrastive Instance Discrimination

Fonseca et al. (2020b), Saeed et al. (2020), and Al-Tahan and Mohsenzadeh (2020) all adapt SimCLR (Chen et al. (2020)) (Section 2.3.1.1) to the audio domain.

[20] That said, many audio deep learning methods do extract feature representations directly from raw waveforms (e.g. Baevski et al. (2020); Hsu et al. (2021)) often through the use of initial one-dimensional convolutions.

[21] However, this is by no means a fully extensive review of audio self-supervised learning methods, since we choose to instead focus on only those works with the highest relevance to our experiments. A full and in-depth analysis on the current SOTA audio self-supervised learning methods can be found in the survey produced by Liu et al. (2022).

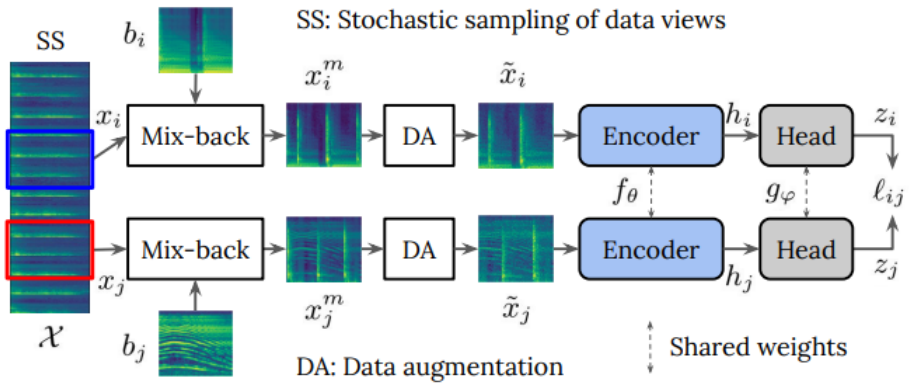


Figure 2.12: **Contrastive Learning of Audio Representations**: The framework for the learning of audio representations using contrastive learning (adapting SimCLR to the audio domain), as implemented by Fonseca et al. (2020b). Figure taken from Fonseca et al. (2020b).

Fonseca et al. (2020b) and Saeed et al. (2020) extract the two views of the audio (processed as a mel-spectrogram), necessary for the two arms of SimCLR's Siamese network, from two different randomly sampled time-frequency (*TF*) patches of the input clip (randomly sampled along the time axis). Fonseca et al. (2020b) additionally perform stochastic application of the data augmentations random resize crop (RRC) and Gaussian noise addition. They further propose another augmentation, which they term *mix-back*, where the incoming TF patch is mixed with another clip randomly drawn from the training dataset, whilst ensuring that the incoming patch remains dominant (the mixing ratio λ is a small value $<< 1/2$, randomly sampled from a uniform distribution as $\lambda \sim \mathcal{U}(0, 0.05)$). They hypothesise that mix-back creates a natural background signal, maintaining the incoming patch in the foreground and therefore "preserving the relevant semantic information" (Fonseca et al. (2020b)) encoded in the signal. This is said to increase the quality of the learned representations, since mix-back preserves the semantic information contained within each individual view whilst at the same time reducing the mutual information between the two views, and as a result the contrastive loss provides a stronger signal for learning.

Al-Tahan and Mohsenzadeh (2020) present **CLAR** (Contrastive L of Auditory Representations), which generates the two views through stochastic application of distortions directly on raw waveforms, considering frequency distortions such as pitch shift, fade in and out, and noise injection, as well as temporal distortions such as time shift, time stretch, and time masking. CLAR further makes use of a supervised cross-entropy loss, which is jointly optimized with the self-supervised contrastive NT-Xent objective (Section 2.3.1.1).

2.4.2.2 (Audio) Non-contrastive Instance Discrimination

Niizumi et al. (2021) present **BYOL-A** (BYOL for Audio), adapting BYOL (Grill et al. (2020)) to the audio domain with minimal modifications from the original learning framework, using audio data preprocessed as mel-spectrograms. The key modification they make is their proposed data augmentation module used to generate the two spectrogram views, which consists of four blocks: pre-normalisation, mix-back (which Niizumi et al. (2021) refer to as *mixup*), RRC, and post-normalisation. BYOL-A also makes use of a lightweight convolutional encoder architecture based on a network used in the solution of the NTT DCASE2020 Challenge Task 6 (Automated Audio Captioning) (Koizumi et al. (2020)). We use this in our experiments, labelling it the *AudioNTT* encoder (Section 3.3.1). Recently, BYOL-A has been extended (**BYOL-A v2**) (Niizumi et al. (2022a)) to include an additional random linear fader (RLF) data augmentation, which acts directly on mel-spectrogram inputs to approximate fade in/out, as well as implementing improvements to the AudioNTT encoder. The BYOL-A augmentation module and AudioNTT encoder are extensively used in our experiments and so, to avoid repetition, we provide limited details here and full explanations in Sections 3.3.1 and 3.2.

Chapter 3

Audio Barlow Twins

The overall research goal of this project is to learn, in a fully unsupervised manner, robust feature representations for audio which are able to generalise well to a broad range of audio downstream tasks with minimal modification[1]. To this end, we present ***Audio Barlow Twins***, adapting the Barlow Twins (Zbontar et al. (2021)) self-supervised learning method from CV to the audio domain. To test the generalisability of the learned features, we evaluate on 18 tasks from the HEAR Challenge (Turian et al. (2022)).

3.1 Learning Framework

As shown in Figure 3.1, the Audio Barlow Twins learning framework mirrors that of other self-supervised learning instance discrimination approaches (Section 2.3.1), using a Siamese network to compute a similarity metric between two distorted views of an input data sample. Unlike in e.g. BYOL (Grill et al. (2020)), where one arm of the Siamese network (the target encoder, or the teacher (in the language of Caron et al. (2021))) is the EMA of the other (the online encoder (student)), Barlow Twins processes each view with the same encoder network parameterised by the same weights θ, avoiding representational collapse not through asymmetric learning updates but instead through a redundancy reduction objective.

1. Generation of views: Audio Barlow Twins first produces two views, v, v' of an input spectrogram[2] x by stochastic application of the Audio Augmentation (AA) module,

$$v, v' \sim \text{AA}(x). \tag{3.1}$$

   The audio augmentation module is largely inspired by the ones proposed in BYOL-A (Niizumi et al. (2021)) and BYOL-A v2 (Niizumi et al. (2022a)), and is discussed in greater detail shortly in Section 3.2

2. Extraction of representations: The two views are then passed through the encoder $f_\theta(\cdot)$ to obtain the representations[3] y_θ, y'_θ,

$$y_\theta = f_\theta(v), y'_\theta = f_\theta(v'). \tag{3.2}$$

   After self-supervised pre-training, the representations y_θ, y'_θ are the ones used for downstream tasks, as shown by the dotted arrow in Fig.3.1.

[1] Through only training a small attached MLP on top of the frozen learned representations (linear evaluation).

[2] All raw waveform are first preprocessed into (log-scaled) mel-spectrograms before applying any data augmentations. This is found to be necessary, despite that the desired effects of the audio augmentations are only approximate when acting on spectrograms and not raw waveforms (Section 3.2), due to an I/O bottleneck observed when directly load in raw waveforms (.WAV files). We overcome this issue by first pre-converting all raw waveforms into mel-spectrograms, and loading them in directly (saved as .npy files) during training.

[3] We are borrowing the language used in the original Barlow Twins publication (Zbontar et al. (2021)), labelling the outputs of the encoder the *representations* and the outputs of the projector, on which the Barlow Twins objective is applied, the *embeddings*.

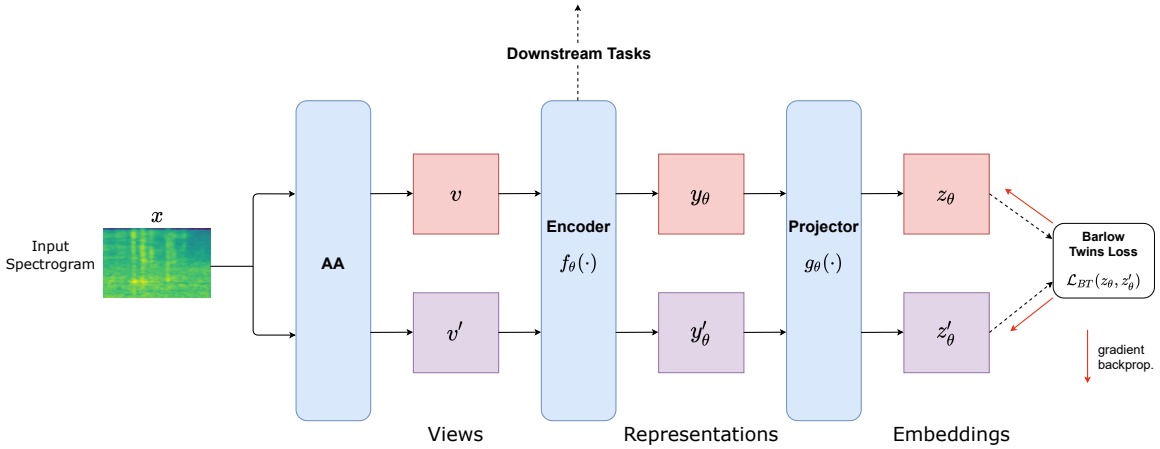


Figure 3.1: **Audio Barlow Twins**: The Audio Barlow Twins learning framework, adapting Barlow Twins (Zbontar et al. (2021)) to the audio domain. Stochastic application of the audio augmentation (AA) module generates two spectrogram views. The representations y_θ, y'_θ for each view are extracted from the encoder, which are then input into the projector to obtain the embeddings z_θ, z'_θ, on which the Barlow Twins objective is applied.

3. Extraction of embeddings: The representations are passed through the projector network $g_\theta(\cdot)$ to obtain the embeddings z_θ, z'_θ,
$$z_\theta = g_\theta(y_\theta), z'_\theta = g_\theta(y'_\theta). \tag{3.3}$$

4. Barlow Twins objective: The Barlow Twins objective, $\mathcal{L}_{BT}$, is then calculated on the embeddings, $\mathcal{L}_{BT}(z_\theta, z'_\theta)$.

   $\mathcal{L}_{BT}$, since it uses batch statistics in its calculation of the embeddings' cross-correlation matrix C, cannot in practice be calculated on an input-by-input basis, but instead must be calculated over a batch of embeddings Z_θ, Z'_θ, with $Z_\theta = [z^1_\theta, ..., z^B_\theta] \in \mathbb{R}^{B\times d}$, and likewise for Z'_θ. Formally:
$$\mathcal{L}_{BT} = \alpha\mathcal{L}^{INV}_{BT} + \lambda\mathcal{L}^{RR}_{BT}, \tag{3.4}$$
   where $\mathcal{L}^{INV}_{BT}$ is the invariance term,
$$\mathcal{L}^{INV}_{BT} = \sum_i (1 - C_{ii})^2, \tag{3.5}$$
   and $\mathcal{L}^{RR}_{BT}$ is the redundancy reduction term,
$$\mathcal{L}^{RR}_{BT} = \sum_{i\neq j} C^2_{ij}. \tag{3.6}$$
   The positive constants α and λ control the trade-off between the importance of the invariance and redundancy reduction terms in the loss, respectively. By default, α is set to 1 and λ is set to 0.005, as in the original publication[4] (Zbontar et al. (2021)). Ablation studies, where the values of α and λ are varied, are performed in Section 4.3.

The cross-correlation matrix C is computed between the embeddings within the batch B,
$$C_{ij} = \sum_{b=1}^{B} \hat{Z}^b_{\theta,i} \hat{Z'}^b_{\theta,j}, \tag{3.7}$$

[4]Tsai et al. (2021) provide a simple argument for setting the value of λ. The first term in the Barlow Twins loss (Eq.3.4) contains d terms and the second term $d(d-1)$ terms, where d is the dimension of the embeddings. Therefore, to correctly balance these terms (with $\alpha = 1$), λ should be set to $\frac{d}{d(d-1)} \approx \frac{1}{d}$. With $d = 8192$, as by default in the original publication (Zbontar et al. (2021)), this would set $\lambda = \frac{1}{8192} \approx 0.0001$, which is 50-fold smaller than the tuned value of 0.005. Zbontar et al. (2021) find that a smaller value of $\lambda << 0.005$ degrades performance (Fig.5 in Zbontar et al. (2021)), alluding that the justification of setting λ to $\frac{1}{d}$ is overly simplistic.

where $\hat{Z}_\theta$ is the normalised embedding Z_θ along the batch dimension, i.e. $\hat{Z}_\theta = \frac{Z_\theta - \mu_{Z_\theta}}{\sigma_{Z_\theta}}$, and $\hat{Z}^b_{\theta,i}$ corresponds to the i^{th} component of the b^{th} batch element of $\hat{Z}_\theta$. The components of C vary between 1 (perfect correlation) and -1 (perfect anti-correlation).

The first term in the loss, $\mathcal{L}^{INV}_{BT}$, forces the on-diagonal terms of the cross-correlation matrix, C_{ii}, to 1, enforcing representational invariance to the applied audio augmentations. The second term in the loss, $\mathcal{L}^{RR}_{BT}$, forces the off-diagonal terms of the cross-correlation matrix, $C_{ij}, i \neq j$, to 0, minimising the redundancy between the individual components of the embeddings, thereby directly preventing representational collapse.

3.2 Audio Augmentations

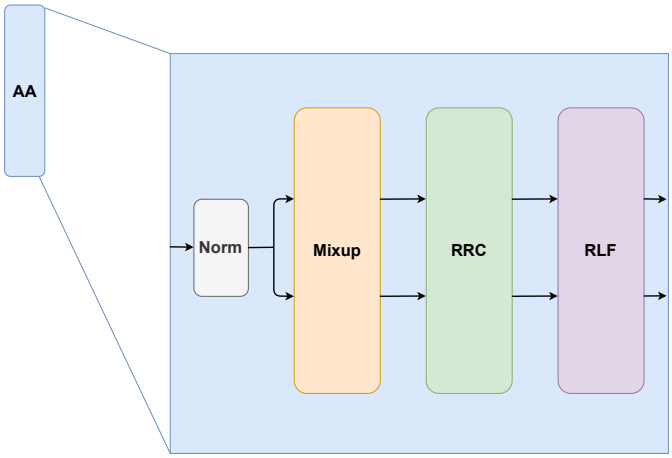


Figure 3.2: **AA Module**: The audio augmentation (AA) module consists of three main components: Mixup, Random Resize Crop (RRC), and Random Linear Fader (RLF). The *Norm* block normalises input spectrograms by the dataset mean and standard deviation. The AA module is borrowed from BYOL-A (Niizumi et al. (2021, 2022a)).

As discussed in Section 3.1, the two spectrogram views are produced through stochastic application of the audio augmentation (AA) module. The audio augmentation module is largely borrowed from BYOL-A (Niizumi et al. (2021, 2022a)), and contains three different augmentation blocks, as shown in Figure 3.2: Mixup (Section 3.2.1), Random Resize Crop (RRC, Section 3.2.2), and Random Linear Fader (RLF, Section 3.2.3). The spectrogram input is first normalised by the dataset mean and standard deviation, $x = (x - \mu_D)/\sigma_D$, as shown by the *Norm* block in Fig.3.2.

3.2.1 Mixup

Similar to *mix-back*, proposed by Fonseca et al. (2020b) (Section 2.4.2.1), Mixup combines the incoming spectrogram, x, with a background signal, x'. The background signal is randomly sampled from a large memory bank of previously encountered inputs, implemented as a first-in-first-out (FIFO) queue. As in BYOL-A (Niizumi et al. (2021)), the memory bank is maintained with fixed size of 2048, where each encountered input is stored (simulatenously removing the oldest element) before being processed by the Mixup block. The randomness of the memory bank is guaranteed by the randomness of the mini-batch shuffled sampling by the `PyTorch` dataloaders.

Formally,

$$\text{Mixup}(x; FIFO, \lambda) = \log((1-\lambda)\exp(x) + \lambda \exp(x')), \tag{3.8}$$

where $x' \sim FIFO(\{x_{i,\text{prev}}\}_{i=1}^{2048})$, and the mixing ratio λ is randomly sampled from a uniform distribution, $\lambda \sim U(0, \alpha)$. The hyperparameter α controls the degree of mixing between the incoming signal, x, and the background signal, x'. With a large value close to 0.5, the mixing can be strong, such that

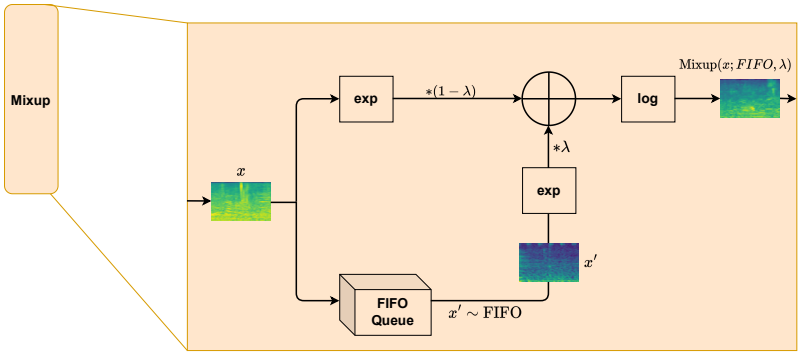


Figure 3.3: **Mixup**: Mixup interpolates an incoming spectrogram with a natural background signal, randomly sampled from a large memory bank of previously encountered inputs. The mixing ratio λ is randomly sampled from a uniform distribution, $\lambda \sim U(0, \alpha)$. In our experiments we find that a value $\alpha = 0.2$ is optimal, allowing the incoming signal to remain dominant above the added background.

the incoming signal will become difficult to perceive above the added background. A smaller value close to 0, however, ensures that the incoming spectrogram remains the dominant signal, whereas a value above 0.5 allows the background to become dominant. In our experiments, we find that (Section 4.1.1) the quality of the learned representations shows high sensitivity to the exact value of α, preferring a small value of $\alpha < 0.5$ (such that incoming signal remains dominant). $\alpha = 0.2$ appears to be optimal[5], and as such this value is used in all further experimentation. We follow the convention set by Fonseca et al. (2020b) and Niizumi et al. (2021), inverting the log in the mel-scale[6] (Eqn.2.21) before mixing is applied, then transforming the output back to a log-scale.

The motivation behind Mixup is two-fold. First, recall that the Audio Barlow Twins learning framework directly enforces representational invariance to the applied audio augmentations through $\mathcal{L}_{BT}^{INV}$ (Eqn.3.5). Therefore, through mixing with background, the representations are encouraged to become robust to noisy signals and more readily discern foreground events, improving representation quality. We anticipate this to be of particular importance in tasks such as speaker recognition. Second, as argued by Fonseca et al. (2020b) and discussed in Section 2.4.2.1, mixing with a random natural background signal whilst maintaining the original signal's dominance (through using a mixing ratio $\lambda < 0.5$), preserves the signal's semantic content whilst simultaneously decreasing the mutual information shared between the two views, producing a stronger signal for learning.

3.2.2 RRC

Random Resize Crop (RRC), a commonly used data augmentation in CV, is implemented as an augmentation applied directly onto the spectrogram to approximate a pitch shift and time stretch of raw waveforms, encouraging the model to learn representational invariance to speed and timbre perturbations of audio.

The RRC procedure follows that exactly as in BYOL-A (Niizumi et al. (2021)), depicted in Figure 3.4. With an input of size $[F, T]$ (F frequency bins, T time bins), a virtual crop boundary is formed of

[5]Mixing hyperparameter $\alpha = 0.2$ is the value also used by BYOL-A v2 (Niizumi et al. (2022a)). BYOL-A (Niizumi et al. (2021)) uses the slightly larger value $\alpha = 0.4$.

[6]Note the flexibility in the language used in this section. Often, we refer to a spectrogram when, to be precise, we mean a (log-scaled) mel-spectrogram. In this report, whenever an input is labelled as a spectogram, we are indeed referring to the (log-scaled) mel-spectrogram.

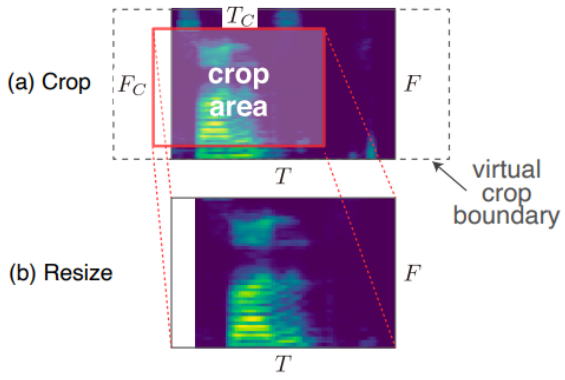


Figure 3.4: **RRC**: Random Resize Crop (RRC) is applied as an approximation to pitch shift and time stretch of raw waveforms. There are two stages to RRC: **(a)** Crop, where a crop area is randomly sampled from within the virtual crop boundary, and **b)** Resize, where the crop area is resized with bicubic interpolation back to the original input size. Figure taken from Niizumi et al. (2022a).

size $[V_F F, V_T T]$, with scaling constants $V_F, V_T > 1$ such that the crop can extend outside of the input (all areas outside are filled with zeros). An area within the virtual crop boundary is then randomly sampled, with frequency span

$$F_C = \lfloor h \times F \rfloor. \tag{3.9}$$

The fractional crop height $h = F_C/F \sim \min(U(f_1, f_2), 1.0)$, where the $\min(\cdot)$ ensures that the sampled crop cannot entirely lie outside of the input's frequency bins (possible otherwise since $V_F > 1$) (Niizumi et al. (2021)), and $[f_1, f_2]$ constitute the range in crop heights (frequency bins) considered. The crop time span is randomly sampled as

$$T_C = \lfloor w \times T \rfloor, \tag{3.10}$$

with fractional crop width $w = T_C/T \sim U(t_1, t_2)$, where $[t_1, t_2]$ constitute the range of crop widths (time frames) considered. The crop region, with size $[F_C, T_C]$, is subsequently randomly sampled from within the virtual crop boundary and resized back to the size of the input ($[F, T]$) with bicubic interpolation. The frequency bin and time frame crop ranges ($[f_1, f_2], [t_1, t_2]$ respectively) are both set as $[0.6, 1.5]$, following Niizumi et al. (2021).

3.2.3 RLF

The Random Linear Fader (RLF) audio augmentation, first proposed by Niizumi et al. (2021), approximates a fade in/out of raw waveforms. $RLF(t)$ is added to each frequency-time bin of an incoming spectrogram x (size $[F, T]$), varying linearly in time. This transforms the input following

$$x_{f,t} \xrightarrow{RLF} x'_{f,t} = x_{f,t} + RLF(t), \tag{3.11}$$

with the addition applied across the entire frequency span of the spectrogram $f = 0, ..., F-1$, for each given time frame $t = 0, ..., T-1$. The linear $RLF(t)$ function is parameterised by two randomly sampled variables $g_0, g_{T-1} \sim U(-1.0, 1.0)$[7],

$$RLF(t) = g_0 + (g_T - g_0)\frac{t}{T}, \tag{3.12}$$

$t = 1, ..., T$, such that g_0 controls the gain applied at the start frame, and g_{T-1} the end frame. Evidently, if $g_T > g_0$, the amplitude of the signal will increase (linearly) with time, approximating an audio source moving towards the listener (fade in), and vice versa for $g_T < g_0$, approximating an audio source moving away from the listener (fade out).

[7]Sampling from $U(-1.0, 1.0)$ allows for interpolation of frequency-time bins with values of the same scale, since the incoming spectrograms are standardised to be distributed $\sim \mathcal{N}(0, 1)$ through normalisation by the dataset mean and standard deviation.

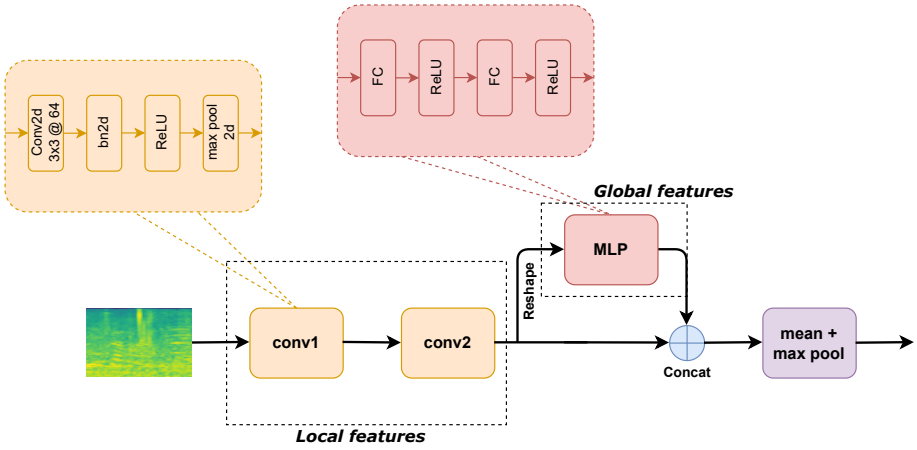


Figure 3.5: **AudioNTT encoder**: Schematic depicting the AudioNTT encoder, proposed by Niizumi et al. (2022a). The AudioNTT encoder consists of two convolutional blocks and a small MLP. Each convolutional block consists of a stride-1 3×3 convolution with 64 output channels, as well as a BatchNorm, ReLU, and stride-2 2×2 max pool. The convolutional blocks extract *local* features from the input spectrograms. The spectrogram outputs of shape $[B, C, F, T]$ are reshaped to $[B, T, D]$, $D = CF$, and fed into a small MLP with one hidden layer, which has both hidden and output dimension 2048. The MLP extracts *global* features. The reshaped local and global features are then concatenated, and a *temporal mean + max pooling* operation is applied. The AudioNTT encoder outputs 3072-*d* features

Table 3.1: **Convolutional encoders**: We consider three convolutional encoders: AudioNTT, ResNet-18, and ResNet-18 ReGP NRF. The flops (G) and throughput for each model are all measured using the `DeepSpeed` (Rasley et al. (2020)) flop profiler with the same Tesla T4 15GB GPU and a batch size of 128 (we report flop count divided by batch size, such that it refers to the flops for the forward pass of one input of size 64×96).

Model	Output size (d)	Params (M)	flops (G)	throughput
AudioNTT	3072	6.33	0.4	265 (inputs/s)
ResNet-18	512	11.2	0.5	390 (inputs/s)
ResNet-18 ReGP NRF	4096	11.2	2.4	80 (inputs/s)

3.3 Encoders

We consider several different architectures for the encoder, $f_\theta(\cdot)$. Namely, the *AudioNTT* convolutional encoder (Koizumi et al. (2020)), as proposed by Niizumi et al. (2022a), ResNet-18 (He et al. (2015)), the Vision Transformer (ViT) (Dosovitskiy et al. (2020)), as well as the ViT$_C$ (Xiao et al. (2021)), a ViT variant shown to have improved performance and training stability (Section 2.2.2.5).

3.3.1 Convolutional Encoders

AudioNTT

The AudioNTT encoder, originally adapted from a solution to the NTT DCASE2020 Challenge Task 6 (Automated Audio Captioning) (Koizumi et al. (2020)), is borrowed from the modified design proposed in BYOL-A v2 (Niizumi et al. (2022a)). A schematic of the AudioNTT encoder is shown in Figure 3.5. The encoder consists of two convolutional blocks and a small MLP. Each convolutional block contains a stride-1 3×3 filter with 64 output channels, followed by BatchNorm, ReLU and a stride-2 2×2 max pool. Each block halves the frequency and time resolutions of the input. Niizumi et al. (2022a) found that adding a third convolutional block, despite increasing the model capacity, harms the quality of the learned audio representations[8] (after BYOL-A (Niizumi et al. (2021)) pretraining). The MLP contains one hidden layer with hidden (and output) dimension 2048, processing the reshaped[9] local features extracted from the convolutional blocks to aggregate global feature infor-

[8] Niizumi et al. (2022a) argue that this is because a third convolutional block halves the time and frequency resolution of the intermediate features and increases the model's maximum receptive field (RF). Koutini et al. (2021) found that CNNs with an "excessive RF" overfit to training data and struggle to generalise for audio tasks, and a narrower RF is preferred.

[9] The frequency embeddings from each channel are concatenated, reshaping the input following $[B, C, F, T] \rightarrow [B, T, D]$, $D = CF$, before input into the MLP.

mation. A skip connection concatenates the reshaped local features and extracted global features, and the final output feature is obtained through *temporal mean + max pooling* (Niizumi et al. (2022a)). For input spectrograms with 64 mel-spaced frequency bins, the AudioNTT encoder produces features with dimensionality 3072. Niizumi et al. (2022a) performed extensive ablation studies on the individual components of the AudioNTT architecture and found that the aforementioned design choices lead to optimal model performance. Full details can be found in the original publication (Niizumi et al. (2022a)).

ResNet-18
We consider a ResNet-18 (He et al. (2015)) encoder, changing the number of input channels in the first convolution from 3 to 1 and removing the final fully-connected (FC) layer. By default, we use the ResNet-C modification as suggested by He et al. (2018), changing the single stride-2 7×7 convolution in the ResNet stem to a series of three 3×3 convolutions with strides $2, 1, 1$ respectively. This modification is found to give a slight performance improvement[10] with minimal additional flops (He et al. (2018)). ResNet-18 outputs 512-*d* features. Further, in addition to the ResNet-18, we consider the ***ResNet-18 ReGP Narrow RF*** modification proposed by Niizumi et al. (2022a). The global average pooling block (before the FC layer) is removed (Remove Global average Pooling) and replaced with the AudioNTT reshaping and mean + max pooling operations, such that the frequency information available in the features is better utilised. The frequency Receptive Field (RF) is narrowed, increasing the features' frequency resolution, through changing the ResNet strides from $[2, 1, 2, 2, 2]$ to $[1, 1, 2, 2, [1, 2]]$. In turn, this increases the representation size by a factor of 8 to 4096. Reducing the model's frequency RF is motivated by work carried out by Koutini et al. (2021), who found that CNNs with an "excessive RF" overfit to training data and struggle to generalise for audio tasks, and a narrower RF is preferred.

3.3.2 Transformer Encoders

The AudioNTT encoder, whilst being highly optimized by Niizumi et al. (2022a) to extract high quality audio representations, has many prior architectural design choices imposed upon it (such as concatenation of local and global features, and temporal mean + max pooling). As a result, we hypothesise that whilst this leads to initial rapid learning of good audio representations, this (in combination with the hard geometric inductive biases inate to all CNNs (Section 2.2.1)) provides a ceiling to the representational capacity of the model, since a structure is preimposed upon the data and not permitted to be fully learned. (Vision) Transformers (Vaswani et al. (2017); Dosovitskiy et al. (2020)), on the other hand, have weak inductive bias, and in addition have immense representational capacity which scales with model size and training length (Chen et al. (2021)). Recent successes in using Vision Transformer (ViT) encoders for self-supervised learning (Caron et al. (2021); Assran et al. (2022); Zhou et al. (2021)) (Section 2.3.1.3) further motivate the use of a ViT encoder.

A schematic for the ViT encoder is shown in Figure 3.6. We use a standard ViT (Dosovitskiy et al. (2020)) architecture. For an input X of size $[B, 1, F, T]$, a series of non-overlapping $p_f \times p_t$ patches[11] with embedding size d are first extracted using a single stride-(p_f, p_t) $p_f \times p_t$ convolution with d output channels. This produces a sequence of length $N = TF/p_f p_t$, with a frequency resolution of F/p_f and a temporal resolution of T/p_t. Following from the recommendation of Chen et al. (2021) to improve training stability, the randomly initialised stride-(p_f, p_t) $p_f \times p_t$ convolution is frozen during training. A learnable `[CLS]` token is prepended to the sequence of patches for aggregation of global information from the entire sequence (Devlin et al. (2018)), and fixed sinusoidal positional encodings are added to each patch (including the `[CLS]` token) to inject positional information into the encoder. The sequence of $N + 1$ tokens is then fed through the L Transformer layers, each of which include a multi-head attention operation with h heads. The output embedding of the `[CLS]` token, $O_{\texttt{[CLS]}}$, is taken as the output representation from the ViT encoder (y_θ in Fig.3.1).

[10] The ResNet-C modification to a ResNet-50 leads to a 0.3% improvmenet in ImageNet validation accuracy while increasing the flops by only $\sim 13\%$ from 3.8G to 4.3G (Table 5 in He et al. (2018)).

[11] Unlike in the original ViT implementation (Dosovitskiy et al. (2020)), we also consider non-square patches.

Table 3.2: **Convolutional stem**: Convolutional stem design details for the ViT$_C$-*B* encoder with different patch sizes. Consistent with the design proposed by Xiao et al. (2021), all 3×3 convolutions are followed by a batch norm and a ReLU, and all models have a final 1×1 convolution to match the Transformer input dimension d (768 for ViT$_C$-*B*). To demonstrate the impact on model complexity, we show the flops (G) and throughput for each patch size, all measured using the `DeepSpeed` (Rasley et al. (2020)) flop profiler with the same Tesla T4 15GB GPU and a batch size of 128 (we report flop count divided by batch size, such that it refers to the flops for the forward pass of one input of size 64×96).

Patch Size	$\#\ 3 \times 3$ convs	output channels	strides	flops (G)	throughput
16×16	4	$[96, 192, 384, 768]$	$[2, 2, 2, 2]$	4.3	60 (inputs/s)
16×8	4	$[96, 192, 384, 768]$	$[2, 2, 2, [2, 1]]$	8.2	35 (inputs/s)
64×2	6	$[96, 192, 384, 768, 768, 768]$	$[2, [2, 1], [2, 1], [2, 1], [2, 1], [2, 1]]$	11.0	25 (inputs/s)
8×8	4	$[96, 192, 384, 768]$	$[2, 2, 2, 1]$	16.0	15 (inputs/s)

We consider the ViT-***B***(ase), ViT-***S***(mall), and ViT-***T***(iny) models, which each contain $L = 12$ Transformer layers with $h = 12, 6$, and 3 heads and embedding sizes of $d = 768, 384$, and 192, respectively. We consider frequency-time patches of shape 16×16, 16×8, and 8×8, which, with 64×96 spectrogram inputs, corresponds to 24, 48, and 96 input patches, respectively. We further consider splitting the inputs into patches of size 64×2 (giving a total of 48 patches), where each patch now corresponds to (two) spectrogram time frames, which brings Audio Barlow Twins more in line with methods considered in prior works such as Liu et al. (2020), which similarly process the spectrogram as a series of time frames in temporal order using a ViT encoder.

In addition, we consider replacing the stride-(p_f, p_t) $p_f \times p_t$ convolution in the stem (the *patchify stem* (Xiao et al. (2021))) with a series of 3×3 convolutions (with strides 1 or 2) followed by a final 1×1 convolution to match the Transformer embedding dimension, d. Xiao et al. (2021) coined this the *convolutional stem*, with the resulting modified ViT labelled the ***ViT***$_C$ (Section 2.2.2.5). The ViT$_C$ shows faster and more stable training than the ViT and improves model performance (Xiao et al. (2021)). Identical to with the vanilla ViTs, we consider the ViT$_C$-***B***(ase), ViT$_C$-***S***(mall), and ViT$_C$-***T***(iny) models[12], where each model corresponds exactly to its vanilla ViT equivalent except for the replacement of the patchify stem with the convolutional stem and the removal[13] of one Transformer layer ($L = 11$).

The design of the ViT$_C$ convolutional stem is dependent on the patch size. As with the vanilla ViT, we consider frequency-time patches of shape 16×16, 16×8, 8×8, and 64×2. For all patch sizes, the convolutional stem contains 4 3×3 convolutions (followed by the final 1×1 convolution), except for with a patch size of 64×2, which contains 6 3×3 convolutions. The stride of each convolution is also dependent on the patch size. The convolutional stem details for the different patch sizes for the ViT$_C$-*B* model can be found in Table 3.2. We note that using a smaller patch size increases the number of tokens on which multi-head self-attention is performed, and therefore increases the model capacity. There exists a trade-off, however, as this comes at the cost of a large increase in flops, reducing model throughput and significantly increasing runtime.

[12] ViT$_C$-*T*, ViT$_C$-*S*, ViT$_C$-*B* correspond to the ViT$_C$-1GF, ViT$_C$-4GF, ViT$_C$-18GF models in the original publication (Xiao et al. (2021)). We change the notation for ease of comparison with the vanilla ViTs.

[13] This exactly follows the architectural design proposed by Xiao et al. (2021), where one Transformer layer is removed to balance the rise in flops due to the increase in the number of convolutions introduced by the convolutional stem.

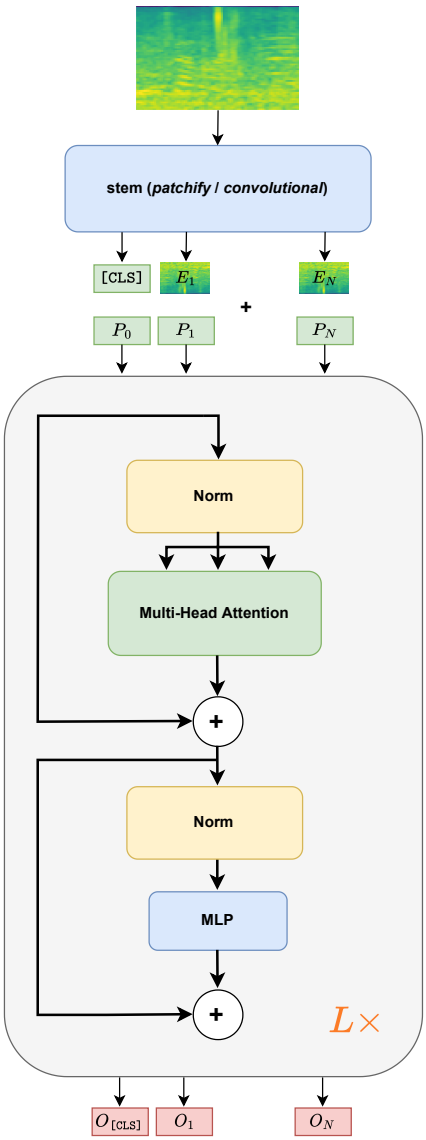


Figure 3.6: **ViT Encoder**: We consider both the ViT and ViT$_C$ encoders. The ViT encoder uses a single stride-(p_f, p_t) $p_f \times p_t$ convolution (*patchify* stem) to extract the $p_f \times p_t$ spectrogram frequency-time patches, whereas the ViT$_C$ uses a stack of ~ 5 3×3 convolutions (*convolutional* stem) followed by a single 1×1 convolution. A learnable `[CLS]` token is prepended to the input sequence, and fixed sinuosidal positional encodings $\{P_0, ..., P_N\}$ are added to each patch. The output embedding of the `[CLS]` token, $O_{[CLS]}$, is taken as the representation that is used in pre-training and for downstream tasks.

3.4 Datasets

3.4.1 AudioSet

Table 3.3: **AudioSet**: Statistics for the number of clips successfully downloaded from the AudioSet dataset (Gemmeke et al. (2017)). AudioSet is split into three subsets, balanced train, unbalanced train, and evaluation.

	Balanced Train	Unbalanced Train	Evaluation
Original dataset	$22,176$	$2,041,792$	$20,383$
Downloaded	$17,783$	$1,629,756$	$16,661$
Download Success Rate	80.2%	79.8%	81.7%

AudioSet (Gemmeke et al. (2017)) is a large-scale audio dataset comprising over 2 million 10 second audio clips, each with associated human-annotations. AudioSet is a multi-label[14] dataset covering a total of 527 sound categories, including speech, music, animal sounds, and common environmental sounds (e.g. echo, reverberation, background noise).

As shown in Table 3.3, AudioSet is split into three subsets, balanced train, unbalanced train, and evaluation. Since the dataset is not available online in an easily downloadable format (due to changes in YouTube's privacy policy) and not already available in storage on departmental servers, to obtain the raw audio clips (.WAV files) we have to extract every clip from calls to YouTube's API. This process takes over a week (both downloading and verifying the quality of the downloaded clips). However, we believe it to be a useful and worthwhile allocation of resources since:

1. AudioSet covers an incredibly large selection of different types of sounds. No other audio dataset contains both the same huge diversity of audio categories and large number of instances per class.

2. Previous works in audio self-supervised learning pre-train on AudioSet (e.g. Niizumi et al. (2021); Saeed et al. (2020); Gong et al. (2021b); Niizumi et al. (2022b); Huang et al. (2022)). Therefore, we similarly wish to pre-train on AudioSet for fair comparison of the robustness of Audio Barlow Twins for learning universal audio representations.

3. Taking the time to download AudioSet and make it available on the departmental servers is potentially of great use to other researchers at Imperial in future studies.

Due to videos being removed or taken down from YouTube since the time of publishing of AudioSet in 2017, not all of the original audio clips are available. We successfully download $17,783$ (80.2%), $1,629,756$ (79.8%), and $16,661$ (81.7%) clips for the balanced train, unbalanced train, and evaluation subsets, respectively. This is lower than that recorded in previous studies (e.g. Gong et al. (2021a)), and as a result we choose to not evaluate our models on the AudioSet evaluation set, since our downloaded subset would not constitute a fair comparison. Instead, we use the large unbalanced train subset ($\sim$ 1.6M instances, corresponding to $\sim$ $4,500$ hours of audio) for self-supervised pre-training.

3.4.2 HEAR Tasks

We use 18 tasks from the HEAR (Holistic Evaluation of Audio Representations) 2021 Challenge (Turian et al. (2022)), derived from 15 datasets, to evaluate the quality of the learned audio representations. HEAR includes two types of tasks:

1. **Scene-based** (Table 3.4), corresponding to classification (multi-class or multi-label) of an entire audio clip.

2. **Timestamp-based** (Table 3.5), corresponding to sound event detection and or transcription over time.

[14] Since AudioSet is multi-label, classification becomes a multi-label classification problem, and as such metrics such as mean average precision (mAP) and area under receiver operator curve (AUROC) are used instead of accuracy.

Table 3.4: **HEAR scene-based tasks**: Scene-based tasks from the HEAR Challenge (Turian et al. (2022)). clf.: classification. Table adapted from Elbanna et al. (2022) and Turian et al. (2022).

Dataset	Task	# Classes	Metric	# Clips	Clip duration (s)	Total duration (h)
Speech Tasks						
CREMA-D (Cao et al. (2014))	Emotion recognition	6	acc	7438	5.0	5.3
LibriCount (Stöter et al. (2018))	Speaker count estimation	11	acc	5720	5.0	7.9
Speech Commands 5h/full (Warden (2018))	Keyword spotting	12	acc	22890/100503	1.0	6.4/27.9
Vocal Imitations (Kim and Pardo (2018))	Vocal imitation clf.	302	mAP	5601	11.26	17.5
VoxLingua (Valk and Alumäe (2020))	Language identification	10	acc	972	18.64	5.0
Environmental Sound Tasks						
ESC-50 (Piczak (2015))	Environmental sound clf.	50	acc	200	5.0	2.8
FSD50K (Fonseca et al. (2020a))	Broad-domain multi-label	200	mAP	51185	$0.3 - 30.0$	108.3
Gunshot triangulation (Cooper and Shaw (2020))	Gunfire location clf.	4	acc	88	1.5	0.04
Music Tasks						
Beijing Opera (Tian et al. (2014))	Instrument clf.	4	acc	236	4.77	0.31
GTZAN (Genre) (Tzanetakis and Cook (2002))	Genre clf.	10	acc	1000	30.0	8.3
GTZAN (Music/Speech)	Music vs speech binary clf.	2	acc	128	30.0	1.07
Mridangam (Anantapadmanabhan et al. (2014))	Stroke & tonic clf.	10 & 6	acc	6977	0.81	1.6
NSynth 5h/50h (Engel et al. (2017))	Pitch clf.	88	acc	5000/49060	4.0	5.6/54.5

Table 3.5: **HEAR timestamp-based tasks**: Timestamp-based tasks from the HEAR Challenge (Turian et al. (2022)). Table adapted from Elbanna et al. (2022) and Turian et al. (2022).

Dataset	Task	Metric	# Clips	Clip duration (s)	Total duration (h)
MAESTRO (Hawthorne et al. (2018))	Music transcription	Onset FMS	185	120.0	6.2
DCASE 2016 Task 2 (Mesaros et al. (2018))	Office sound detection	Onset FMS	72	120.0	2.4

Scene-based tasks, following Elbanna et al. (2022), can be subdivided into three subcategories: speech, environmental sounds, and music. A lightweight version of the HEAR Challenge (which we label **HEAR-*L***) is used for performance comparison in our Audio Barlow Twins ablation studies (Section 4.3). HEAR-*L* consists of five tasks covering all three of the scene-based task subcategories: CREMA-D (speech), LibriCount (speech), FSD50K (environmental[15] sounds), ESC-50 (environmental sounds), and GTZAN Genre (music).

All datasets are downloaded[16] at 48 kHz and re-sampled to 16 kHz to align with the sampling frequency of the AudioSet clips used during model pre-training. The re-sampling is implemented using the `librosa` Python library (McFee et al. (2015)).

For each task, the embeddings are first extracted from the frozen pre-trained model. The timestamp-based tasks first require the input audio clips to be divided into fixed-size segments, such that embeddings can be extracted corresponding to specific timestamps. We use a segment size of 950 ms with a hop size of 50 ms for all timestamp-based tasks. The embeddings are then evaluated using the `hear-eval`[17] toolkit, which trains a shallow MLP classifier on the frozen embeddings (linear evaluation). The MLP is trained for a maximum of 500 epochs with the Adam optimizer (Kingma and Ba (2014)), implementing early stopping on the validation set, checking every 3 epochs with a patience of 20 (except for with DCASE 2016 Task 2, which is checked every 10 epochs). Model selection is performed over a choice of 8 models each of which uses a different hyperparameter configuration, selecting the optimal model using the validation score. Variations in the number of hidden layers, learning rate, and weight initialization are considered. Full details can be found in the original HEAR publication (Turian et al. (2022)).

3.4.2.1 FSD50K

We give special attention to the FSD50K dataset (FreeSound Dataset 50k) (Fonseca et al. (2020a)), one of the 15 HEAR datasets, since we use this dataset for hyperparameter tuning (Section 4.1.1) and ablation studies (Section 4.3). FSD50K is a multi-label dataset containing a total of $51,197$ audio clips covering 200 sound categories, which are derived from a subset of the AudioSet Ontology. The audio clips vary in length between 0.3 and 30s. FSD50K is split into two subsets, a development subset

[15] FSD50K (Fonseca et al. (2020a)) is included in the environmental sounds subcategory, following Elbanna et al. (2022), although it also contains speech and music.

[16] `https://zenodo.org/record/5887964`

[17] `https://github.com/hearbenchmark/hear-eval-kit`

($40,966$ clips, average clip length of 7.1s) and an evaluation subset ($10,231$ clips, average clip length of 9.8s). A further train/validation is split is provided for the development subset.

Chapter 4

Experiments

4.1 Experimental Setup

We conduct large-scale self-supervised pre-training using the Audio Barlow Twins learning framework. We pre-train on the unbalanced train subset of AudioSet ($\sim$ 1.6M audio segments) for 100 epochs with a batch size of 128, which corresponds to $\sim$ 1.3M training iterations. We consider all three convolutional encoders, as well as the ViT$_C$(-***B***,-***S***, and -***T***) encoder with a patch size of 16×8. We choose to not pre-train on AudioSet with the vanilla ViTs due to limited resources, since we find that the ViT$_C$ encoders significantly outperform the vanilla ViTs (Section 4.4.1) and full AudioSet pre-training with a Transformer encoder takes several days to complete[1].

Audio preprocessing All audio samples are converted from raw .WAV files, with a sampling frequency of 16 kHz, to (log-scaled) mel-spectrograms using a 64 ms sliding window with a 10 ms step size, extracting $F = 64$ mel frequency bins in the range $60 - 7,800$ Hz[2]. By default, during pre-training we randomly crop $T = 96$ time frames (all clips with shorter duration are padded with zeros), corresponding to 950 ms of audio. This produces a mel-spectrogram of size $F \times T = 64 \times 96$. These choices exactly follow those made in BYOL-A (Niizumi et al. (2021)).

Audio Barlow Twins architecture We use a smaller version of the projector network than that proposed in the original Barlow Twins publication (Zbontar et al. (2021)), although with the same modular structure. The projector network corresponds to a small MLP with one hidden layer, which has hidden dimension 8192, and an output dimension of 1048. The first layer of the projector is followed by a batch normalisation and ReLU non-linearity. The Barlow Twins loss hyperparameters are set to 1 and 5×10^{-3}, for α and λ respectively.

Optimization The choice of optimizer varies dependent on the encoder architecture. For all optimizers, we linearly warm up the learning rate during the first epoch, after which we decay it using a cosine schedule (Loshchilov and Hutter (2016)).

i. *Convolutional encoders* -
We use the LARS optimizer (Layer-wise Adaptive Rate Scaling) (You et al. (2017)), with a learning rate of 0.4 for the weights and 0.0048 for the biases and batch normalisation parameters. We use a weight decay of $1 \cdot 10^{-5}$. Following from Zbontar et al. (2021), LARS adaptation, as well as weight decay, do not apply to the biases and batch normalisation parameters. The choice of the LARS optimizer, over Adam and SGD, as well as the optimizer hyperparameters (learning rate weights, learning rate biases, weight decay) are selected after conducting an extensive hyperpa-

[1] Audio Barlow Twins pre-training on the unbalanced train subset of AudioSet with a ViT$_C$-*B* on a single NVIDIA RTX 6000 GPU, using mixed precision, takes approximately 120 hours.

[2] Note the upper limit of $7,800$ Hz, just below the Nyquist frequency f_{nq} of 8 kHz for a 16 kHz sampling frequency (Section 2.4.1).

rameter sweep (Section 4.1.1). We also consider the default values[3] as used by Zbontar et al. (2021), but find a noticeable degradation in model performance.

ii. *ViT$_C$ encoders* -
We use AdamW with the default hyperparameter values as suggested[4] by Xiao et al. (2021), using a learning rate of $6.25 \cdot 10^{-5}$ and a weight decay of 0.24. Following Xiao et al. (2021), weight decay is not applied to the biases and any normalisation parameters. AdamW's β parameters are set as $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Audio augmentation parameters We use Mixup with $\alpha = 0.2$ (Section 3.2.1). For RRC, we use a virtual crop boundary of size $[F, 1.5T]$, where the input spectrogram has size $[F, T]$. These values are all chosen from an initial hyperparameter sweep (Section 4.1.1). The audio augmentations are applied directly to the preprocessed mel-spectrograms.

4.1.1 Hyperparameter Tuning

Before performing full Audio Barlow Twins pre-training on AudioSet, hyperparameter sweeps are conducted over several important variables that we anticipate will most significantly affect optimization. We consider these to be the choice of optimizer, the optimizer learning rate (*lr*) and weight decay (*wd*), as well as the hyperparameters associated with the Audio Barlow Twins audio augmentation (AA) module (Section 3.2), namely the Mixup ratio α and the virtual crop boundary scaling constants V_F and V_T.

For each hyperparameter configuration, Audio Barlow Twins pre-training is performed on the FSD50K development subset for a maximum of 20 epochs, corresponding to ~ 6400 training iterations (with a batch size of 128). We measure performance through training a linear classifier[5] (linear evaluation protocol) on the frozen features of the FSD50K train subset, extracted from the pre-trained model, implementing early stopping on the FSD50K validation subset with a patience of 10. The classifier is trained for a maximum of 100 epochs. We report the model performance as the mAP on the FSD50K evaluation subset. We anticipate this metric to be generally indicative of the quality of the learned audio representations for a given set of hyperparameter values. During linear evaluation on FSD50K, all audio clips are first randomly cropped to 96 time frames, the same number as used during Audio Barlow Twins pre-training. This is done to reduce the time to extract the embeddings from the pre-trained model. This explains the discrepancy between the FSD50K test scores (mAP) reported here and in the later sections, where linear evaluation is also performed on FSD50K but with either an increased number of crop frames (711, corresponding to the average clip length of 7.1s) or no cropping (HEAR evaluation).

The hyperparameter sweeps are implemented using the `Optuna` framework (Akiba et al. (2019)), evaluating 16 hyperparameter configurations (*trials*) per sweep. Each set of hyperparameter values are sampled each trial using `Optuna`'s TPE sampler (Tree-structured Parzen Estimator), and we evaluate model performance at the end of each of the 20 pre-training epochs to allow pruning[6] of unpromising trials at intermediate stages (before the maximum of 20 epochs). All hyperparameter sweeps use the same audio preprocessing and Audio Barlow Twins architectural defaults as specified in Section 4.1. That is except for the projector output dimension, which has a slightly smaller value of 256[7]. We measure the importance of individual hyperparameters (i.e. optimization sensitivity) for

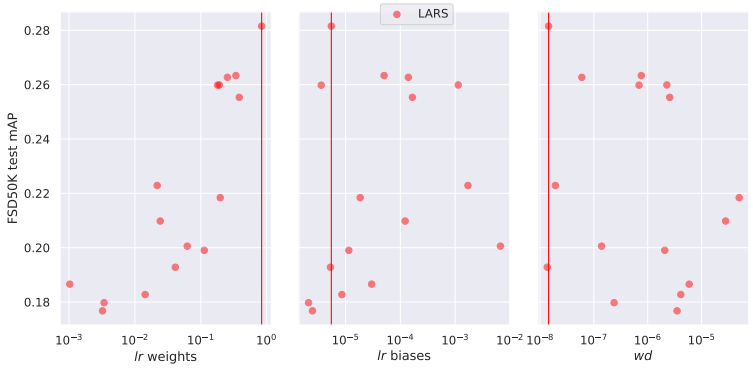
[3] Appropriately scaled for a batch size of 128 (using linear scaling), Zbontar et al. (2021) use a learning rate of 0.1 for the weights and 0.0024 for the biases, with a weight decay of $1.5 \cdot 10^{-6}$.

[4] Xiao et al. (2021) perform an extensive hyperparameter sweep for the ViT$_C$ with a batch size of 2048, using a patch size of 16×16. They find, with AdamW, a *lr* of $1 \cdot 10^{-3}$ and a *wd* of 0.24 to be optimal. We scale this *lr* linearly by batch size $(0.24 \cdot 128/2048)$ to obtain the used value of $6.25 \cdot 10^{-5}$.

[5] We train the linear classifier with the following set of hyperparameters: Adam optimizer, batch size $= 200$, *lr* $= 1 \cdot 10^{-3}$, *wd* $= 1 \cdot 10^{-8}$, Adam $\beta_1 = 0.9$, Adam $\beta_2 = 0.999$, Adam $\epsilon = 1 \cdot 10^{-8}$.

[6] We use `Optuna`'s Hyperband pruner to prune unpromising trials.

[7] We use a projector output dimension of 256 both in the hyperparameter sweeps (Section 4.1.1) and all ablation studies (Section 4.3) as initial experimentation suggests that this value is optimal. Extensive ablation studies, however, reveal that a larger value of 1048 is preferable, and as such this value is used in full AudioSet pre-training.



(a) **LARS**

(b) **Adam, SGD**

Figure 4.1: **AudioNTT optimizer hyperparameter sweep**: Using the AudioNTT encoder, we train 16 models with the LARS, Adam, and SGD optimizers, training each for 20 epochs on the FSD50K development subset. For Adam and SGD, each model uses a different randomly sampled (*lr*, *wd*), considering $lr = \{10^{-6}, 10^{-2}\}$ and $wd = \{10^{-3}, 10^{0}\}$. For LARS, each model uses a different randomly sampled (*lr weights*, *lr biases*, *wd*), considering *lr weights* $= \{10^{-3}, 10^{0}\}$, *lr biases* $= \{10^{-6}, 10^{-2}\}$, and $wd = \{10^{-8}, 10^{-4}\}$. Scatterplots are shown for all three optimizers **(a)** LARS (red), **b)** Adam (green), SGD (blue)), with FSD50K test mAP under linear evaluation shown on the vertical axis. Vertical bars correspond to the optimal values for each optimizer.

a given sweep using the fANOVA[8] hyperparameter importance evaluation algorithm (available within the `Optuna` framework) (Hutter et al. (2014)). fANOVA fits a random forest regression model to the scores of the completed (unpruned) trials for the trial hyperparameter configurations. The variance of the fitted model is then decomposed into additive components, each of which are associated with a specific hyperparameter. The fractional variance associated with a hyperparameter is taken as its importance.

4.1.1.1 Optimizer, Learning Rate and Weight Decay

AudioNTT

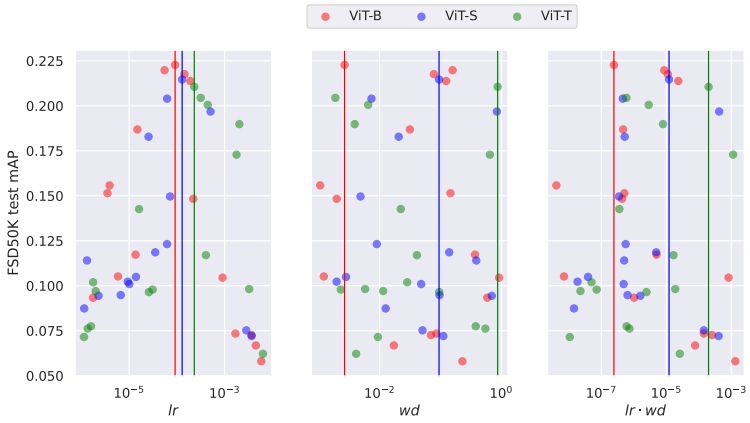
For the AudioNTT encoder, we consider pre-training with Adam, SGD and LARS. For Adam and SGD, each trial a (*lr*, *wd*) pair is sampled (on a log scale), considering $lr = \{10^{-6}, 10^{-2}\}$ and $wd = \{10^{-3}, 10^{0}\}$, with all other optimizer hyperparameters set to their `PyTorch` defaults[9]. Bias and batch normalisation parameters are excluded from weight decay. For LARS[10], we consider separate a *lr* for the weights and for the biases, considering *lr weights* $= \{10^{-3}, 10^{0}\}$, *lr biases* $= \{10^{-6}, 10^{-2}\}$, and $wd = \{10^{-8}, 10^{-4}\}$. A (*lr weights*, *lr biases*, *wd*) triplet is sampled (on a log scale) each trial. For all three optimizers a total of 16 trials are evaluated.

Results Figures 4.1a (LARS) and 4.1b (Adam, SGD) show scatterplots for the FSD50K test scores

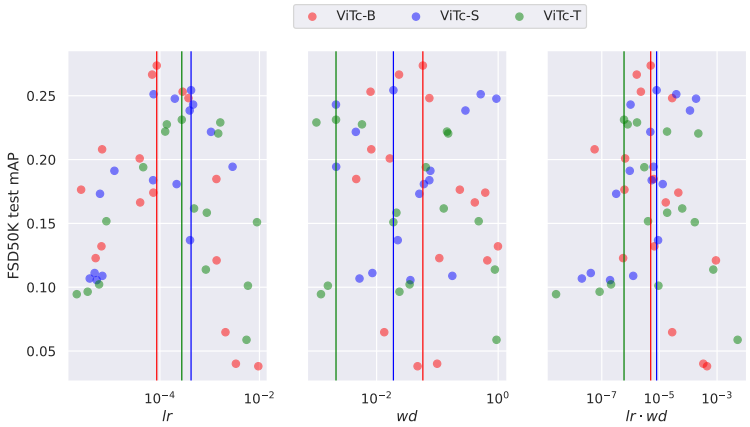
[8]Full details on the fANOVA algorithm can be found in the original publication: Hutter et al. (2014).

[9]For SGD: Nesterov momentum = False. For Adam: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \cdot 10^{-8}$.

[10]The other LARS hyperparameters (see You et al. (2017)) are set as: momentum $m = 0.9$, LARS coefficient $\eta = 1 \cdot 10^{-3}$



(a) **ViT**



(b) **ViT$_C$**

Figure 4.2: **ViT / ViT$_C$ optimizer hyperparameter sweep**: Using the ViT (**a**)) and ViT$_C$ (**b**)) encoders, with a fixed patch size of 16×16, we train 16 models with AdamW, training each for 20 epochs on the FSD50K development subset. Each model uses a different randomly sampled (*lr*, *wd*), considering $lr = \{10^{-6}, 10^{-2}\}$ and $wd = \{10^{-3}, 10^{0}\}$. Scatterplots are shown for all three model sizes (***-B*** (red), ***-S*** (blue), ***-T*** (green)), with FSD50K test mAP under linear evaluation shown on the vertical axis. Vertical bars correspond to the optimal values for each model.

(mAP) for the models trained with the three optimizers. We observe that LARS and Adam both attain higher optimal performance (~ 0.28) than SGD (~ 0.26). LARS is also less sensitive to the *wd* value than Adam and SGD, and as such LARS is used as the default optimizer for pre-training with a convolutional encoder[11]. From Fig.4.1a, we see that LARS tends to prefer a larger value for *lr weights* $\sim O(10^0)$, showing less sensitivity to *lr biases* and *wd*, with importance values of 0.81, 0.10, and 0.09, respectively. The optimal trial, which achieves a score of 0.282, uses (*lr weights*, *lr biases*, *wd*) values of $(0.84, 5.5 \cdot 10^{-6}, 1 \cdot 10^{-8})$. We find in general that (*lr weights*, *lr biases*, *wd*) values of $(0.4, 4.8 \cdot 10^{-3}, 1 \cdot 10^{-6})$ are effective across all convolutional encoders (including ResNet-18, ResNet-18 ReGP Narrow RF), and as such these values are used with the LARS optimizer by default.

ViT / ViT$_C$

For the ViT and ViT$_C$ encoders, we consider pre-training only with AdamW[12]. We use a patch size of 16×16. Each trial, a (*lr*, *wd*) pairs is sampled (on a log scale), considering $lr = \{10^{-6}, 10^{-2}\}$ and

[11]The choice of LARS as the default optimizer (with a convolutional encoder) is further motivated by that LARS is used in the original Barlow Twins publication (Zbontar et al. (2021))

[12]Xiao et al. (2021) find that the ViT$_C$ models are also stable when trained with SGD. However, initial experimentation reveals that training ViT$_C$ encoders with SGD (with Audio Barlow Twins pre-training) is unstable, with the loss frequently going to NaN.

wd $= \{10^{-3}, 10^{0}\}$, with all other hyperparameters set to their `PyTorch` defaults[13]. Bias and batch normalisation parameters are excluded from weight decay. Hyperparameter sweeps are conducted for all ViT and ViT$_C$ model sizes (*-B*, *-S*, *-T*), evaluating a total of 16 trials for each model.
Results Figures 4.2a (ViT) and 4.2b (ViT$_C$) show scatterplots for the FSD50K test scores (mAP) for the ViT and ViT$_C$ models. We observe that for all ViT and ViT$_C$ models, a *lr* of $\sim 1 \cdot 10^{-4}$ is optimal, although the smaller models (*-S*, *-T*) tend to prefer a slightly larger value. The models show less sensitivity to the *wd* value. In general, a *lr/wd* of $1 \cdot 10^{-4}/0.06$ is effective for all ViT and ViT$_C$ models, and as such these values are used by default during the ablation studies (Section 4.3). However, we find that the tuned values used by Xiao et al. (2021) for the ViT$_C$ models, a *lr/wd* of $6.25 \cdot 10^{-5}/0.24$, lead to better model performance[14] when pre-training on the full AudioSet unbalanced train subset. We anticipate that this is because a slightly lower *lr* is preferred for a significantly longer training schedule[15]. We also note three further salient points: **1)** The ViT$_C$ models reach a higher optimal performance than their corresponding vanilla ViT models for all model sizes (*-B*: 0.27 vs 0.22,*-S*: 0.25 vs 0.21, *-T*: 0.23 vs 0.21), **2)** The ViT$_C$ models show less sensitivity to the exact *lr* value used than the ViT models, with wider peaks for all three model sizes, supporting the stability claims made by Xiao et al. (2021), **3)** Both ViT and ViT$_C$ optimal performance scales with model size.

4.1.1.2 Audio Augmentation Hyperparameters

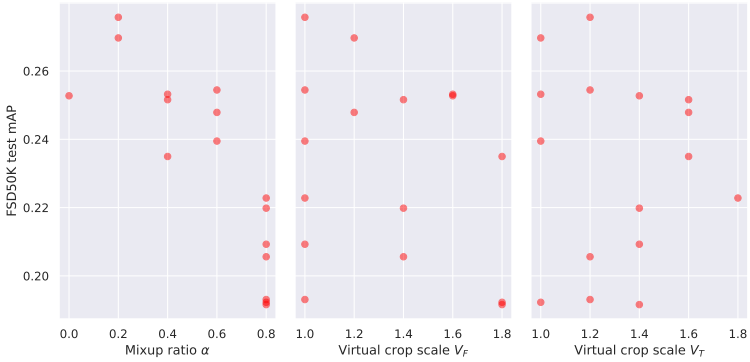


Figure 4.3: **Mixup, virtual crop scale hyperparameter sweep**: Using the AudioNTT encoder, with the LARS optimizer (default *lr weights*, *lr biases*, *wd* values of 0.4, 4.8×10^{-3}, $1 \cdot 10^{-6}$, respectively), we train 16 models for 20 epochs on the FSD50K development subset. Each model uses a different randomly sampled Mixup ratio α and virtual crop boundary scaling constants, V_F and V_T , considering $\alpha = \{0, 0.2, 0.4, 0.6, 0.8\}$ and $V_F, V_T = \{1, 1.2, 1.4, 1.6, 1.8\}$. Scatterplots are shown for all α, V_F, V_T values, with FSD50K test mAP under linear evaluation shown on the vertical axis.

We pre-train with the AudioNTT encoder, using the LARS optimizer with default *lr weights*, *lr biases*, and *wd* values (found in Section 4.1.1.1) of 0.4, 4.8×10^{-3}, $1 \cdot 10^{-6}$. The Mixup ratio α and virtual crop boundary scaling constants, V_F and V_T, are varied each trial, considering $\alpha = \{0, 0.2, 0.4, 0.6, 0.8\}$ and $V_F, V_T = \{1, 1.2, 1.4, 1.6, 1.8\}$. A total of 16 trials are evaluated.
Results The Mixup ratio α is found to be by far the most important of the three hyperparameters, with importance values 0.79, 0.12, and 0.09 for α, V_F, and V_T, respectively. A high value of α is found to consistently lead to poor model performance (left plot, Fig.4.3), which corresponds to when the added background sound dominates over the incoming signal (Section 3.2). $\alpha = 0.2$ is found to be optimal, which is the same value as used by BYOL-A v2 (Niizumi et al. (2022a)). $V_F > 1$ tends, in general, to harm model performance (centre plot, Fig.4.3), and as such a value of $V_F = 1$ is chosen. $V_F = 1$ corresponds to randomly cropped (and then resized) segments in the incoming spectrogram containing no zero padded frequency bins, and as such this indicates that interpolating outside of the original spectrogram frequency range (with zeros) can change the semantic content of the signal,

[13]For AdamW: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \cdot 10^{-8}$
[14]Model performance as measured by tracking FSD50K linear evaluation score every 5 epochs during training.
[15]100 epochs on AudioSet unbalanced train segments corresponds to $\sim 250 \times$ training iterations as with 20 epochs on the FSD50K development subset.

such that embedding the two views near one another becomes harmful to the quality of the learned representations. A slightly larger V_T value is preferred (right plot, Fig.4.3), with $V_T = 1.5$ used by default (since this corresponds with the value used in BYOL-A (Niizumi et al. (2021)) and BYOL-A v2 (Niizumi et al. (2022a))).

4.2 Results

4.2.1 HEAR

We compare the performance of the Audio Barlow Twins pre-trained models on the 18 HEAR tasks with three baseline models. Namely, **CREPE** (Kim et al. (2018)), **wav2vec2.0** (Baevski et al. (2020)), and **BYOL-A*** (Niizumi et al. (2021)). CREPE trains a 1D CNN in a supervised manner for pitch estimation on 16 hours of synthesised music. wav2vec2.0 is self-supervised, jointly optimizing a masked latent reconstruction objective and a contrastive loss on quantized latent representations. wav2vec2.0 extracts the latent representations using a 1D CNN, which acts on raw waveforms, and uses a Transformer to build global context representations, pre-training on $100,000$ hours of speech data. BYOL-A* is a reimplementation of BYOL-A (Niizumi et al. (2021)) by Elbanna et al. (2022), which we use since Niizumi et al. (2021) did not evaluate on the HEAR tasks.

4.2.1.1 Scene-based Tasks

The results on the speech scene-based HEAR tasks can be found Table 4.1, the environmental sound tasks in Table 4.2, and the results on the music tasks in Table 4.3.

Table 4.1: **Results on HEAR speech scene-based tasks**: We report results for all full AudioSet pre-trained models on the HEAR speech scene-based tasks. [HEAR] refers to results from the HEAR baseline models used for comparison, whereas [ABT] refers to models pre-trained using Audio Barlow Twins. All results are obtained using the `hear-eval` toolkit. Top two performing models for each task are shown underlined and highlighted. % $\uparrow_{\text{RAND}}$ refers to the fractional increase (**not** the absolute increase) from the average score obtained by the random baseline (without any pre-training) for that model.

			Speech				
Model	CREMA-D	LbC	SPC-5h	SPC-F	VocIm	VoxL	**Avg** (% $\uparrow_{\text{RAND}}$)
[HEAR] CREPE	0.383	0.499	0.180	0.211	0.051	0.142	0.244
[HEAR] wav2vec2.0	0.656	0.692	0.838	0.879	0.080	0.493	0.606
[HEAR] BYOL-A*	0.623	0.788	0.896	0.924	0.137	0.390	0.626
[ABT] AudioNTT	0.594	0.745	0.882	0.910	0.111	0.324	0.594 (17%)
[ABT] ResNet-18	0.489	0.624	0.336	0.408	0.053	0.166	0.346 (4%)
[ABT] ResNet-18 ReGP NRF	0.541	0.739	0.545	0.641	0.067	0.196	0.455 (27%)
[ABT] ViT$_C$-*B* (16×8)	0.581	0.812	0.724	0.771	0.087	0.312	0.548 (140%)
[ABT] ViT$_C$-*S* (16×8)	0.547	0.769	0.684	0.746	0.095	0.273	0.519 (150%)
[ABT] ViT$_C$-*T* (16×8)	0.535	0.748	0.609	0.674	0.0894	0.263	0.487 (150%)

Table 4.2: **Results on HEAR environmental sound scene-based tasks**: We report results for all full AudioSet pre-trained models on the HEAR environmental sound scene-based tasks.

	Environmental Sound			
Model	ESC-50	FSD50K	Gunshot	**Avg** (% $\uparrow_{\text{RAND}}$)
[HEAR] CREPE	0.301	0.159	0.863	0.441
[HEAR] wav2vec2.0	0.561	0.342	0.848	0.584
[HEAR] BYOL-A*	0.789	0.489	0.875	0.7180
[ABT] AudioNTT	0.786	0.474	0.905	0.721 (24%)
[ABT] ResNet-18	0.537	0.273	0.765	0.525 (38%)
[ABT] ResNet-18 ReGP NRF	0.679	0.358	0.786	0.608 (27%)
[ABT] ViT$_C$-*B* (16×8)	0.705	0.446	0.845	0.666 (49%)
[ABT] ViT$_C$-*S* (16×8)	0.703	0.431	0.860	0.665 (72%)
[ABT] ViT$_C$-*T* (16×8)	0.659	0.397	0.714	0.590 (33%)

4.2.1.2 Timestamp-based Tasks

The results on the timestamp-based HEAR tasks can be found Table 4.4.

Table 4.3: **Results on HEAR music scene-based tasks**: We report results for all full AudioSet pre-trained models on the HEAR music scene-based tasks.

	Music							
Model	Beijing	GTZAN-Genre	GTZAN-M/S	Mrd-Stroke	Mrd-Tonic	NSynth 5h	NSynth 50h	**Avg** (% $\uparrow_{\text{RAND}}$)
[HEAR] CREPE	0.928	0.645	0.929	0.898	0.824	0.870	0.900	0.856
[HEAR] wav2vec2.0	0.907	0.780	0.946	0.943	0.828	0.402	0.653	0.780
[HEAR] BYOL-A*	0.919	0.835	0.969	0.970	0.900	0.290	0.642	0.789
[ABT] AudioNTT	0.966	0.818	0.962	0.970	0.932	0.476	0.740	0.838 (−1%)
[ABT] ResNet-18	0.902	0.656	0.961	0.848	0.768	0.138	0.317	0.647 (−13%)
[ABT] ResNet-18 ReGP NRF	0.924	0.757	0.992	0.933	0.809	0.228	0.408	0.722 (−9%)
[ABT] ViT$_C$-*B* (16 × 8)	0.869	0.765	0.992	0.952	0.897	0.280	0.632	0.769 (15%)
[ABT] ViT$_C$-*S* (16 × 8)	0.826	0.768	0.976	0.943	0.860	0.236	0.567	0.739 (14%)
[ABT] ViT$_C$-*T* (16 × 8)	0.860	0.745	0.969	0.924	0.839	0.192	0.472	0.714 (20%)

Table 4.4: **Results on HEAR timestamp-based tasks**: We report results for all full AudioSet pre-trained models on the HEAR timestamp-based tasks. All models use a frame size of 950ms and a hop size of 50ms. Error rate (↓) indicates that a lower error rate is better. Table format adapted from Elbanna et al. (2022).

	DCASE		MAESTRO		**Avg** (% $\uparrow_{\text{RAND}}$)
Model	Onset FMS	Error rate (↓)	Onset FMS	Onset w/ Offset FMS	Onset FMS
[HEAR] CREPE	0.552	0.420	0.3910	0.15	0.472
[HEAR] wav2vec2.0	0.670	0.320	0.0328	0.009	0.351
[HEAR] BYOL-A*	0.499	0.503	0.0028	0.00029	0.251
[ABT] AudioNTT	0.761	0.274	0.04801	0.00672	0.405 (27%)
[ABT] ResNet-18	0.513	0.524	0.00099	0.00019	0.257 (−11%)
[ABT] ResNet-18 ReGP NRF	0.587	0.394	0.00925	0.00162	0.298 (−20%)
[ABT] ViT$_C$-*B* (16 × 8)	0.722	0.275	0.0263	0.00429	0.374 (−10%)
[ABT] ViT$_C$-*S* (16 × 8)	0.664	0.400	0.0191	0.00278	0.342 (−12%)
[ABT] ViT$_C$-*T* (16 × 8)	0.696	0.350	0.0130	0.00198	0.355 (−3%)

4.2.1.3 Discussion

On a par performance with BYOL-A We observe that Audio Barlow Twins, with the AudioNTT encoder, generally performs on a par with, and in some cases outperforms, BYOL-A*[16], which uses the same AudioNTT encoder architecture, on the scene-based tasks, and consistently outperforms BYOL-A* on the timestamp-based tasks. We see further consistent improvements over the HEAR baseline models (CREPE, wav2vec2.0), except on the type of tasks on which these models have been specialised (music for CREPE, speech for wav2vec2.0). These results, therefore, demonstrate the robustness of the Audio Barlow Twins method in the generation of general-purpose audio representations.

AudioNTT performs best out of the encoders Of the considered encoder architectures, the AudioNTT convolutional encoder is consistently the strongest performer. We note the poor performance of the ResNet-18 encoder[17], although still on several tasks outperforming the HEAR baseline models (CREPE, wav2vec2.0). However, with the minor architectural modifications of the ResNet-18 to the ResNet-18 ReGP NRF, removing the global pooling operation (replacing with reshaping followed by mean + max pooling) and narrowing the model's frequency receptive field, we find significant performance improvements, increasing the average scores on the speech, environmental sound, and music scene-based tasks by 0.11, 0.08, and 0.07, respectively, and by 0.04 on the timestamp-based

[16] Whilst BYOL-A* is similarly trained on AudioSet for 100 epochs, its *actual* amount of training is higher than with our Audio Barlow Twins implementation. This is because: **1)** Elbanna et al. (2022) train on 5,800 hours of data from AudioSet, which is ∼ 1,300 hours more than we use, and as a result their training duration is actually 1.3 × longer. **2)** For each BYOL forward pass, to symmetrise the loss (Eqn.2.16), each view is passed through both arms of the Siamese network. However, since the Barlow Twins learning framework is already symmetric, we do not do this. As a result, BYOL-A* performs two forward passes for every one we do.

[17] Due to the unexpectedly poor performance of the ResNet-18 encoder after 100 epochs of Audio Barlow Twins pre-training on AudioSet, we further investigate whether this is due to the choice of optimization hyperparameters, which are tuned to the AudioNTT and not the Resnet-18 encoder: (*lr weights*, *lr biases*, *wd*) of $(0.4, 0.0048, 1 \cdot 10^{-5})$ with the LARS optimizer. An additional full AudioSet pre-training is run with the ResNet-18 encoder with the default optimization hyperparameters (scaled for a batch size of 128) as used in the original Barlow Twins publication (Zbontar et al. (2021)), that is (*lr weights*, *lr biases*, *wd*) of $(0.1, 0.0024, 1.5 \cdot 10^{-6})$. However, we find that this does not lead to an improvement in model performance, but in fact a slight degradation. We conclude, therefore, that the poor performance of the ResNet-18 encoder is not attributable to the optimization hyperparameters.

tasks. This indicates that the strength of the results with the AudioNTT and ResNet-18 ReGP NRF encoders is partially attributable to the Audio Barlow Twins learning framework, as shown by strong improvements over the randomly initialised models (except for on the music tasks, discussed shortly), but also attributable to their architectural designs which impose strong inductive biases on the audio data. Contrarily, the ViT$_C$-*B* encoder, which imposes no such inductive biases, also achieves strong performance across the HEAR scene-based tasks, showing more significant improvement on its random baseline (e.g. its average score on the speech tasks increases by 140%). This therefore provides further evidence for the effectiveness of the Audio Barlow Twins learning framework. We additionally note that performance scales with model size for the ViT$_C$ encoders, with the ViT$_C$-*B* outperforming the ViT$_C$-*S*, and likewise for the ViT$_C$-*S* and ViT$_C$-*T*.

Audio Barlow Twins pre-training harms performance on music tasks Despite the clear benefits of Audio Barlow Twins pre-training for learning high-quality audio representations, achieving on-par results with BYOL-A*, we note that Audio Barlow Twins pre-training appears damaging, not beneficial, to performance on several of the music tasks, often leading to performance degradation from the random baselines. We find this to be particularly evident for the Mridingam Stroke and Tonic, NSynth (5h and 50h), and MAESTRO tasks. For example, the AudioNTT encoder after Audio Barlow Twins pre-training achieves a pitch classification accuracy of 0.476 on NSynth 5h, but the AudioNTT random baseline scores 0.656. Audio Barlow Twins appears to be damaging to performance on tasks where a sound's pitch is required to be correctly discerned. However, recall that invariance to pitch perturbations is enforced through RRC (Section 3.2.2), and as such, it is intuitive that a model will consequently struggle to classify pitch[18]. As we show in our extensive ablation studies (Section 4.3), RRC does significantly improve the quality of the learned representations on the majority of the HEAR tasks (in fact, on all of the HEAR-*L* tasks). We therefore observe that using instance discrimination approaches to learn universal audio representations is intrinsically flawed, to some extent, since they rely on applying data augmentations to generate two (or more) views, and any given data augmentation may benefit one type of audio task but harm another. This provides support for generative and predictive self-supervised methods for learning universal audio representations (e.g. Gong et al. (2021b); Niizumi et al. (2022b); Huang et al. (2022)), since they don't require the use of any data augmentations. We note that this issue appears not to be limited to only Audio Barlow Twins, as BYOL-A*, which also uses RRC (and the same AudioNTT architecture), performs similarly poorly on the aforementioned HEAR music tasks (in fact considerably worse than (AudioNTT) Audio Barlow Twins).

4.2.2 Performance During Training

To track the quality of the learned representations during Audio Barlow Twins pre-training, we monitor the performance of the models under linear evaluation of FSD50K. FSD50K linear evaluation is performed in the exact same way as detailed in Section 4.1.1, fitting a linear classifier for a maximum of 100 epochs on the frozen features of the FSD50K train subset, with early stopping with a patience of 10 on the FSD50K validation subset, evaluating on the FSD50K evaluation subset. However, to bring the results more in line with those reported by HEAR evaluation, where the input clips aren't cropped, we crop all clips to the FSD50K development subset average duration of 7.1s (whereas in Section 4.1.1 we crop to a duration of 950ms). The training hyperparameters are identical to those reported in Section 4.1.1. For the convolutional encoders (AudioNTT, ResNet-18, ResNet-18 ReGP NRF), we evaluate every 5 epochs during training, and for the ViT$_C$ encoders (-*B*, -*S*, -*T*), we evaluate more frequently, every 2 epochs.

We consider both fitting a linear classifier on all of the features of the FSD50K train subset, which corresponds to using 100% of the provided labels (hence we label this scenario **100%**), as well as

[18]We verify this point empirically. We train two AudioNTT encoders for 100 epochs on the FSDK50K development subset with Audio Barlow Twins (Section 4.3), one with Mixup+RLF+RRC (default audio augmentations), and one without RRC (Mixup+RLF), all other hyperparameters remaining the same. Evaluating on NSynth 5h, we find that Mixup+RLF+RRC achieves an accuracy of 0.646 (below the random baseline), whereas Mixup+RLF scores 0.706. Note that Mixup+RLF+RRC does better on NSynth 5h than that reported in Table 4.3, which is as expected since it has been trained for significantly fewer training iterations (and training with RRC is harmful to performance on this task).

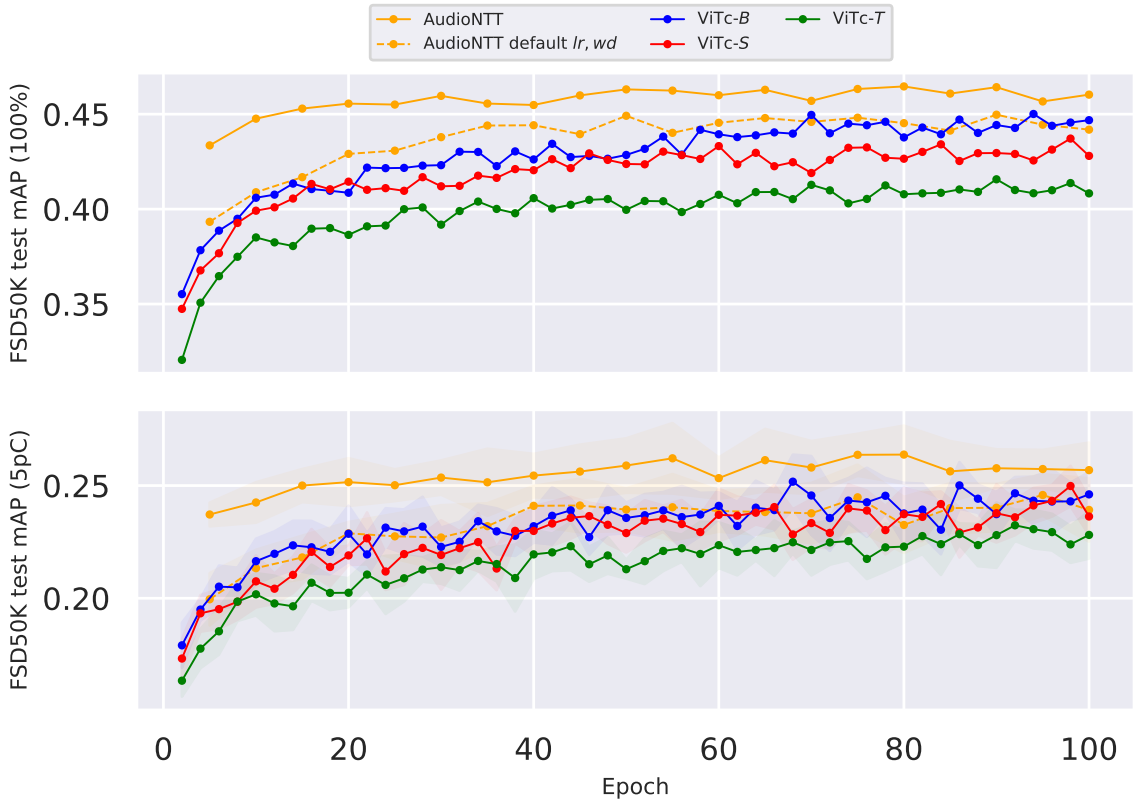


Figure 4.4: **Training curves**: Linear evaluation scores on FSD50K for the AudioNTT (yellow) and ViT$_C$ encoders (*-B* (blue), *-S* (red), *-T* (green)) during full AudioSet Audio Barlow Twins pre-training. We report test mAP when training on **100%** of the labels from the FSD50K train subset (upper plot), as well as the mean score and standard deviation across 3 random splits of the data, using only 5 examples per class (**5pC**) (lower plot). We evaluate every 5 epochs for the AudioNTT encoder and every 2 epochs for the ViT$_C$ encoders. We additionally show the training curve (dashed yellow lines) for the AudioNTT encoder pre-trained using different optimization hyperparameters ((*lr weights*, *lr biases*, *wd*) of $(0.1, 0.0024, 1.5 \cdot 10^{-6})$).

fitting on three randomly selected subsets of the FSD50K train subset, each of which contain exactly 5 audio clips per class, and as such we label this scenario **5pC** (5 per Class). The latter corresponds to low-shot linear evaluation, and measures the model's ability to generalise to new data having previously seen very few labelled examples. For low-shot linear evaluation, we report the mean classification score over the three subsets, alongside the standard deviation. The resulting training curves are shown in Figure 4.4.

We observe that after only a few training epochs, the AudioNTT encoder rapidly reaches near optimal model performance, achieving a FSD50K test mAP of ~ 0.45 (100%, ~ 0.25 for 5pC). After epoch ~ 20, however, the performance of the AudioNTT encoder plateaus. We find that this rapid rise in AudioNTT performance followed by a plateau once the ceiling of FSD50K test mAP ~ 0.45 (100%) has been reached is not attributable to the initial high *lr* of the LARS optimizer (*lr weights* of 0.4), since the dashed yellow line in Fig.4.4 shows AudioNTT training with a lower *lr weights* of 0.1. The dashed yellow line similarly doesn't breach the AudioNTT performance celing of ~ 0.45 (instead it just learns more slowly). Contrarily, the ViT$_C$ encoders optimize much more slowly, with performance plateauing more gradually and FSD50K test score increasing throughout the 100 training epochs, although slowing towards the end (this plateau towards epoch 100 may also be influenced by *lr* decay towards 0). This is further emphasised by the results reported in Table 4.5, showing that whilst AudioNTT performance stagnates, ViT$_C$*-B* performance continues to rise.

We anticipate, therefore, that the strong inductive biases of the AudioNTT encoder result in rapid

Table 4.5: **FSD50K linear evaluation scores during pre-training**: Linear evaluation scores on FSD50K for both the AudioNTT and ViT$_C$-*B* (16×8) encoders at different stages during full AudioSet Audio Barlow Twins pre-training. We report test mAP when training on **100%** of the labels from the FSD50K train subset, as well as the mean score and standard deviation across 3 random splits of the data, using only 5 examples per class (**5pC**). For the 100% scores for both models at epochs 20, 50, and 100, we also show the the fractional increase from the previous entry, e.g. $\uparrow 5.1\%$ for the AudioNTT encoder at epoch 20 refers to that 0.456 is a 5.1% rise from the 100% score at epoch 5, 0.434. This is to give a measure for whether the model performance is plateauing with increased training.

		FSD50K Test mAP	
Epoch	**Architecture**	100%	5pC
5	AudioNTT	0.434	0.237 ± 0.005
	ViT$_C$-*B* (16×8)	0.378	0.195 ± 0.005
20	AudioNTT	0.456 ($\uparrow 5.1\%$)	0.252 ± 0.011
	ViT$_C$-*B* (16×8)	0.409 ($\uparrow 8.2\%$)	0.229 ± 0.012
50	AudioNTT	0.463 ($\uparrow 1.5\%$)	0.259 ± 0.012
	ViT$_C$-*B* (16×8)	0.429 ($\uparrow 4.9\%$)	0.236 ± 0.007
100	AudioNTT	0.464 ($\uparrow 0.2\%$)	0.257 ± 0.012
	ViT$_C$-*B* (16×8)	0.447 ($\uparrow 4.2\%$)	0.246 ± 0.010

optimization and strong model performance. But, these same biases then limit the model's representational capacity, providing a ceiling to the quality of the learned representations. In contrast, the ViT$_C$ encoders, which pre-impose very weak inductive biases, are slower to optimize, since they have to learn the structure of the audio data from scratch, but the performance of the models continues to rise throughout training. We therefore expect that with a longer training schedule, and more training data, the ViT$_C$ models' performances will continue to improve, eventually outperforming the AudioNTT encoder. We attribute this to the immense representational capacity of the ViT, and is consistent with the findings of previous studies (e.g. Dosovitskiy et al. (2020); Chen et al. (2021)). Unfortunately, we are unable to confirm this hypothesis, as longer training schedules are beyond the scope of this project due to limited computational resources[19].

[19] We cannot, for example, train across several GPUs for weeks on end.

4.3 Ablation Studies

We perform extensive ablation studies to investigate the contributions of each of the different components of the Audio Barlow Twins learning framework. For all ablation studies (except for the ViT ablation studies, Section 4.4), we perform Audio Barlow Twins pre-training for 100 epochs with the AudioNTT encoder on the FSD50K development subset, which corresponds to $\sim$ 32k training iterations (with a batch size of 128). Whilst this is a significantly shorter training schedule than full pre-training on AudioSet ($\sim$ 40-fold shorter), the AudioNTT encoder is found to rapidly optimize (Section 4.2.2), and as such performance after 100 epochs of pre-training on FSD50K is expected to be highly correlated with performance after 100 epochs of pre-training on AudioSet (i.e. we would expect to see the same trends). Asides from the training duration and training dataset, all ablation studies use the same experimental set-up as used for full AudioSet pre-training (Section 4.1), except for the projector output dimension, which is set by default to 256. For all ablations considered, we evaluate model performance on the HEAR-*L* tasks under linear evaluation using the `hear-eval` toolkit. HEAR-*L* consists of five HEAR tasks covering all three of the scene-based task subcategories: CREMA-D (speech), LibriCount (speech), FSD50K (environmental sound), ESC-50 (environmental sound), and GTZAN Genre (music).

4.3.1 Audio Augmentations

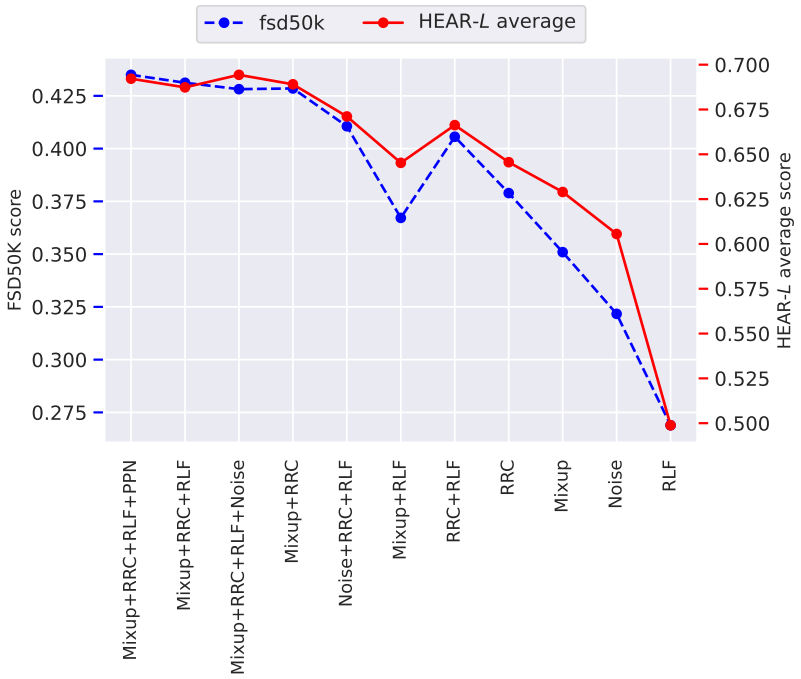


Figure 4.5: **Effect of different audio augmentations**: We compare the effect of pre-training with different combinations of the components of the Audio Barlow Twins audio augmentation (AA) module. Results are shown both evaluated on FSD50K (blue) and the average score on the 5 HEAR-*L* tasks (red).

We consider using different combinations of the components of the audio augmentation (AA) module (Section 3.2). We further consider two different variations, namely Pre-Post-Norm (*PPN*) and *Noise*. PPN refers to removal of the normalisation block, which standardises input spectrograms by the dataset mean and standard deviation, and replacing it with the Pre- and Post-Normalisation blocks proposed by Niizumi et al. (2021) in BYOL-A. Specifically, the pre-normalisation block normalises input spectrograms by batch (and not dataset) statistics, and the post-normalisation block does the same, but after the application of the audio augmentations (Mixup, RRC, RLF). Niizumi et al. (2021)

Table 4.6: **Comparison of audio augmentation ablation studies**: We show the FSD50K test mAP and average score on the HEAR-*L* tasks for different audio augmentation ablation studies, as well as for a randomly initialised model (no pre-training). Also shown is the percentage increase of the HEAR-*L* average score for each ablation study from the only RLF ablation study (**Δ RLF**). The entry for Mixup+RRC+RLF is shown highlighted as this is the set-up used by Audio Barlow Twins by default.

		HEAR-*L*	
Augmentations	**FSD50K (mAP)**	**Avg (%)**	**Δ RLF**
Mixup+RRC+RLF+PPN	0.435	69.2%	+19.2%
Mixup+RRC+RLF+Noise	0.428	69.4%	+19.4%
Mixup+RRC+RLF	0.431	68.7%	+18.7%
Noise+RRC+RLF	0.411	67.1%	+17.1%
RRC	0.379	64.6%	+14.6%
Mixup	0.351	62.9%	+12.9%
Noise	0.322	60.6%	+10.6%
RLF	0.269	50.0%	
Random	0.314	54.1%	+4.1%

argue the post-normalisation corrects the statistical drifts caused by the applied augmentations. Noise refers to the addition of random noise to an incoming spectrogram. We implement this for direct comparison with Mixup, which instead of noise adds a natural background signal randomly sampled from the training dataset. We sample the noise from a Gaussian distribution $\mathcal{N}(0, \lambda)$, where $\lambda \sim U(0, \alpha)$ as with Mixup[20].

From Figure 4.5 and Table 4.6, we note four salient points.

1. Strong audio augmentations are essential to learn high-quality representations
When all the augmentations are removed from the baseline except RLF (remove RRC and Mixup), Audio Barlow Twins performance drops significantly, by 19 points from 69% to 50% average on the HEAR-*L* tasks.

2. Mixing with natural background signals is more effective than with noise
Mixup improves 13 points from the RLF average on HEAR-*L* to 63%, whereas addition of Gaussian noise results in a smaller improvement of only 10%. Further, using Noise as well as Mixup+RRC+RLF leads to no significant additional performance improvements.

3. RRC is the most effective audio augmentation
RRC, which approximates pitch shift and time stretch, attains the highest performance when any of the audio augmentations are applied alone, achieving a HEAR-*L* average score of 65% (compared with 63% for Mixup, 60% for Noise, and 50% for RLF). RLF is by far the least effective augmentation. These findings are consistent with previous results found by Niizumi et al. (2022a).

4. PPN shows minimal improvement over dataset normalisation
Mixup+RRC+RLF+PPN results in almost identical model performance as Mixup+RRC+RLF (with normalisation by dataset statistics), both having a HEAR-*L* average score of $\sim 69\%$.

4.3.2 Batch Size

Figure 4.6a shows the sensitivity of Audio Barlow Twins to batch size. The size of the batch is expected to influence training, and therefore downstream performance, since batch dynamics contribute significantly to the Barlow Twins objective (Eqn.3.4) through the empirical cross-correlation

[20]Sampling noise in this way allows for the magnitude of the values that are added to the incoming spectrogram to be of the same scale as they are when using Mixup, since with Mixup the randomly sampled mixing spectrogram x' has first been normalised. This therefore allows for fair comparison of the two techniques.

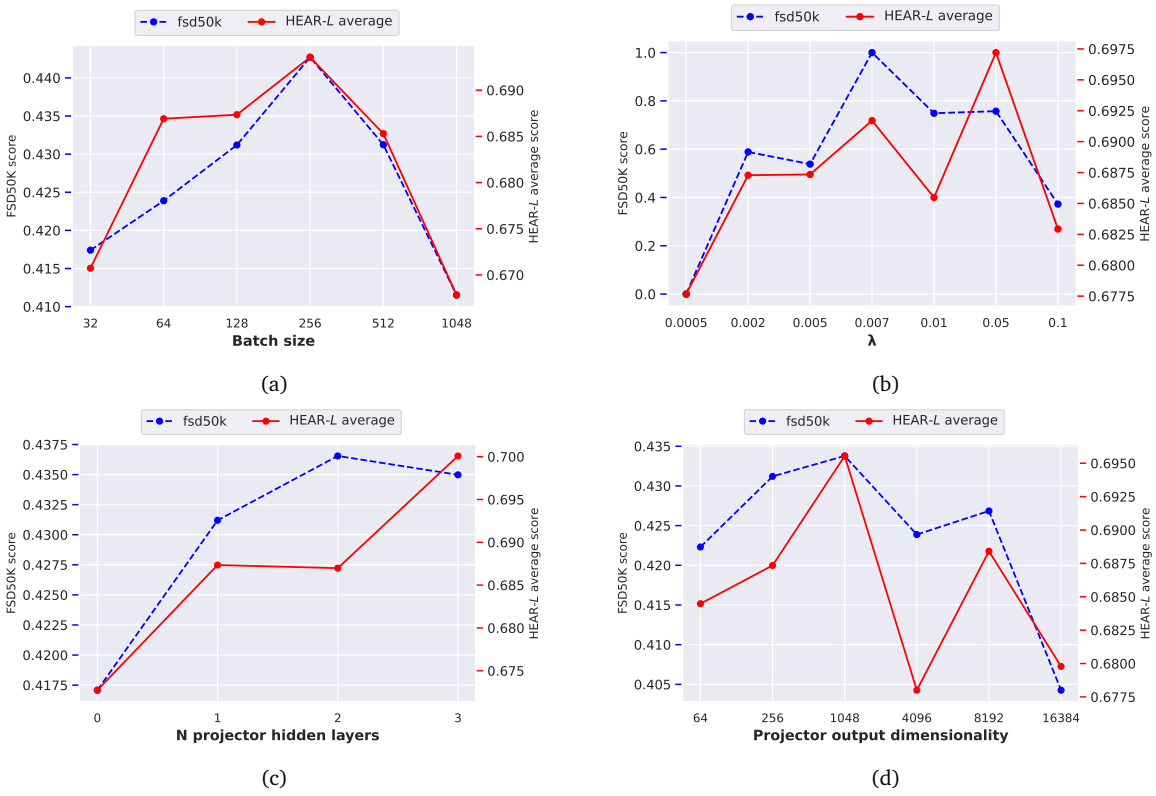
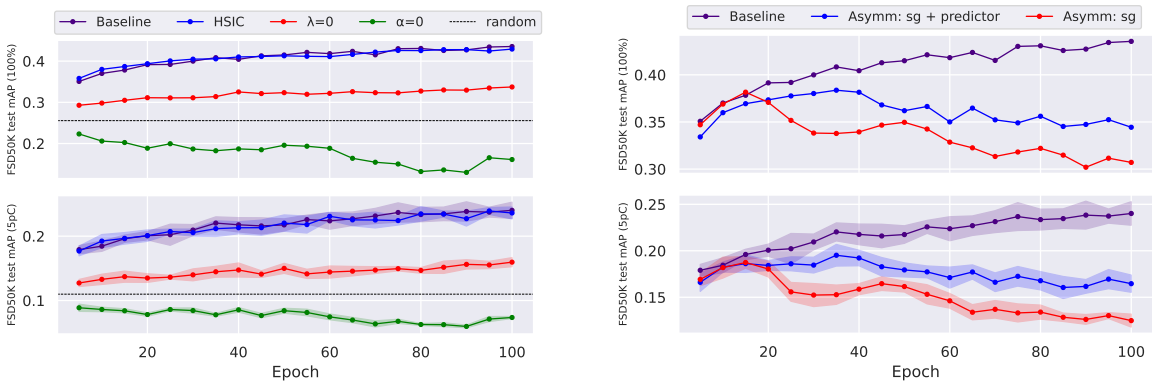


Figure 4.6: **Ablation studies**: Ablation studies for **(a)** batch size, **(b)** Barlow Twins objective hyperparameter λ, **(c)** projector depth, and **(d)** projector output dimensionality. Results are shown both evaluated on FSD50K (red) and the average score on the 5 HEAR-L tasks (blue).

matrix, which is computed across the batch embeddings (Eqn.3.7: $C_{ij} = \sum_{b=1}^{B} \hat{Z}^b_{\theta,i} \hat{Z'}^b_{\theta,j}$). We observe that Audio Barlow Twins exhibits reasonable sensitivity to batch size, with strongest performance with a medium-sized batch containing $64 - 512$ samples. This is contrary to methods in CV such as SimCLR (Chen et al. (2020)), which prefer much larger batch sizes (SimCLR requires a batch size of at least 1048 for strong performance). However, we note that all models are trained with the learning rates (*lr weights* and *lr biases*) tuned with a batch size of 128, applying linear scaling: $lr = lr_{128} \times \text{BatchSize}/128$. Re-tuning LARS learning rates for each batch size is beyond the scope of this project. We are unable to consider batch sizes above 1048 due to GPU memory restrictions.

4.3.3 λ, Projector Depth, Projector Output Dimensionality

Figures 4.6b, 4.6c, and 4.6d show the variation of model performance with the Barlow Twins objective hyperparameter λ, the number of hidden dimensions of the projector network, and the dimensionality of the embeddings (over which the Barlow Twins objective is calculated). We observe that Audio Barlow Twins shows minimal sensitivity to the exact value of λ, as found by Zbontar et al. (2021) in the original Barlow Twins publication, although a value in the approximate range $0.002 < \lambda < 0.05$ is preferred, allowing for both the invariance and redundancy reduction terms of the Barlow Twins objective (Eqn.3.4) to contribute. We further observe that a deeper projector is preferred, although performance does not significantly rise above a depth of 2 (1 hidden layer). Contrary to the observations of Zbontar et al. (2021), we don't find that model performance continues to improve as projector output dimensionality grows, with saturation at an output size of 1048, and significant performance degradation observed with a dimensionality of $16,384$.



(a) (b)

Figure 4.7: **Ablation studies**: Training curves for **(a)** Barlow Twins objective ablation studies, and **(b)** asymmetric learning updates ablation studies.

4.3.4 Barlow Twins Objective Function

We implement three modifications to the Barlow Twins objective function (Eqn.3.4). **1)** $\alpha = 0$, such that $\mathcal{L}_{BT} = \mathcal{L}_{BT}^{RR}$ (Eqn.3.6), and as a result only the redundancy reduction objective is enforced. **2)** $\lambda = 0$, such that $\mathcal{L}_{BT} = \mathcal{L}_{BT}^{INV}$ (Eqn.3.5), and as a result only the invariance objective is enforced. **3)** The Barlow Twins loss is modified to the the Barlow Twins *HSIC* loss ($\mathcal{L}_{HSIC}$), proposed by Tsai et al. (2021), who derive a relationship between the Barlow Twins objective and the Hilbert-Schmidt Independence Criterion (Gretton et al. (2005)), with the slight modification to the objective function that the off-diagonal terms of the cross-correlation matrix C are encouraged towards -1 (perfect anti-correlation), instead of towards 0.

Figure 4.7a shows the training curves of the different Barlow Twins objective function ablations, monitoring FSD50K test mAP (100% and 5pC), identically to as in Section 4.2.2. **1)** Setting $\alpha = 0$, such that the model only learns to minimise the redundancy between the individual components of the embeddings, leads to very poor performance, with training harming the quality of the learned representations (resulting in performance far below the random baseline). **2)** Setting $\lambda = 0$ also significantly harms performance below the Barlow Twins baseline ($\alpha = 1, \lambda = 5 \cdot 10^{-3}$), although the model is able to learn some useful information through enforcing the invariances to the applied data augmentations. It is clear, however, as similarly reported by the original Barlow Twins authors (Zbontar et al. (2021)), that using both the invariance and redundancy reduction terms together is required for strong solutions. **3)** The alternative HSIC loss, $\mathcal{L}_{HSIC}$, leads to near identical performance to the Barlow Twins baseline.

4.3.5 Asymmetric Learning Updates

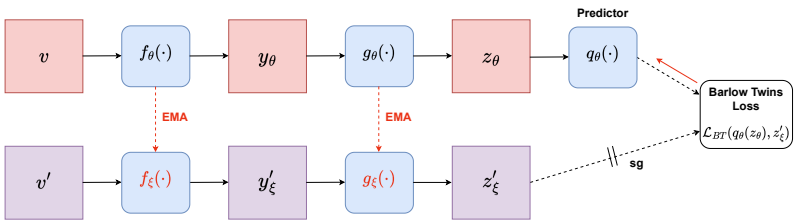


Figure 4.8: **Asymmetric Audio Barlow Twins**: The Audio Barlow Twins learning framework modified to have an asymmetric structure, closely mirroring approaches such as BYOL (Grill et al. (2020)). The weights of the target network, ξ, evolve as the EMA of the online networks', θ. The Barlow Twins objective is calculated over the outputs of the predictor, which acts the online embeddings, and the target embeddings, $\mathcal{L}_{BT}(q_\theta(z_\theta), z_\xi)$.

(Audio) Barlow Twins avoids representational collapse by design, through redundancy reduction.

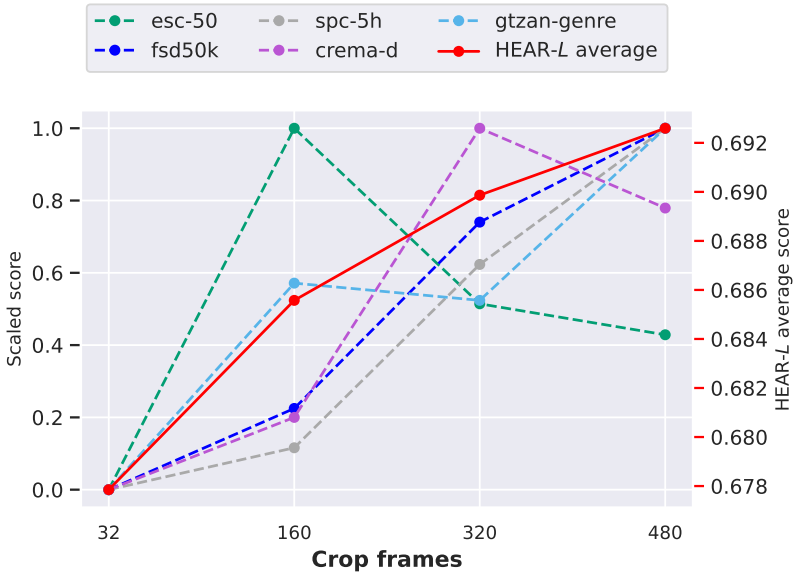


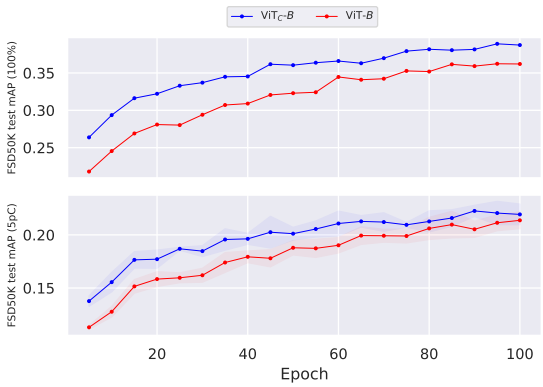
Figure 4.9: **Effect of the input audio duration**: We compare the effect of pre-training with a different length of the input audio used during pre-training, considering cropping the input spectrograms to 32, 160, 320, and 480 frames. Results are shown evaluated on the individual HEAR-*L* tasks (ESC-50 (green), Speech Commands 5h (grey), GTZAN Genre (light blue), FSD50K (dark blue), CREMA-D (purple)), as well as the HEAR-*L* average (red). Since the scores on the individual HEAR-*L* tasks have different scales, we show the MinMax scaled scores, where the best performing input length for each task is set to 1 and the worst performing to 0.

However, as discussed in detail in previous sections, many SOTA self-supervised learning approaches implement asymmetric student-teacher learning frameworks (e.g. BYOL (Grill et al. (2020)), DINO (Caron et al. (2021)), MSN (Assran et al. (2022))). We consider, therefore, if introducing asymmetry into Audio Barlow Twins leads to an improvement in the quality of the learned representations. To do this, we modify the Audio Barlow Twins learning framework to mirror that of BYOL (as shown by the schematic in Figure 4.8), introducing an online and target network, where the weights of the target network, ξ, are the EMA of the online networks', θ. We consider learning both with and without an additional predictor network. Clearly, as shown in Figure 4.7b, this results in highly unstable learning, with significantly reduced model performance. This suggests, therefore, that using the Barlow Twins objective alongside a student-teacher asymmetric learning framework results in unstable learning dynamics. However, we note that we do not re-tune any hyperparameters for this ablation study, and so it remains a possibility that stable learning can be achieved with careful hyperparameter selection. This is out of the scope of this project, but remains an interesting research direction for future studies.

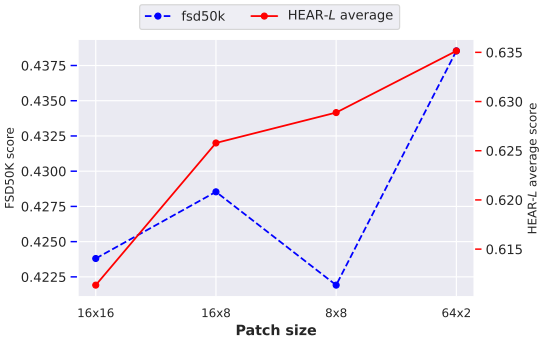
4.3.6 Input Audio Duration

We consider variations in the length of the input audio used during Audio Barlow Twins pre-training. Specifically, we consider cropping the input spectrograms to 32, 160, 320, and 480 frames, which correspond to $\sim$ 320ms, 1.6s, 3.2s, and 4.8s of audio, respectively. We choose not to show the results with the default 96 crop frames as all hyperparameters have been tuned using this value, and as a result it is not considered to be a fair comparison. As shown in Figure 4.9, almost all HEAR-*L* tasks benefit from a longer training window, with the exception of ESC-50. For speech tasks, such as Speech Commands or CREMA-D, this is intuitive, since an element of speech may last several seconds, and as such using only a short segment of under one second in pre-training can result in sounds that don't retain the original semantic content of the clip (e.g. a word may be cropped to only a syllable or single character). However, ESC-50 clearly seems to benefit from a shorter input duration, showing optimal performance with 160 input frames. The environmental sound dataset ESC-50 contains many sounds categories which consist of short, sharp noises, such as the categories mouse click and door knock, and as such using a short segment during pre-training may better align with the actual sound

duration of this dataset. The optimal input duration during pre-training therefore appears to be dependent on the downstream dataset being evaluated on, although there exists a general trend that longer clips are beneficial.



(a) **ViT vs ViT$_C$**: Training curves comparing the ViT-*B* and ViT$_C$-*B* encoders. Both are trained a patch size of 16×16.



(b) **Effect of patch size**: We compare the effects of pre-training with the ViT$_C$ encoder using different patch sizes. Results are shown both evaluated on FSD50K (blue) and the average score on the 5 HEAR-*L* tasks (red).

Figure 4.10

4.4 ViT / ViT$_C$ Ablation Studies

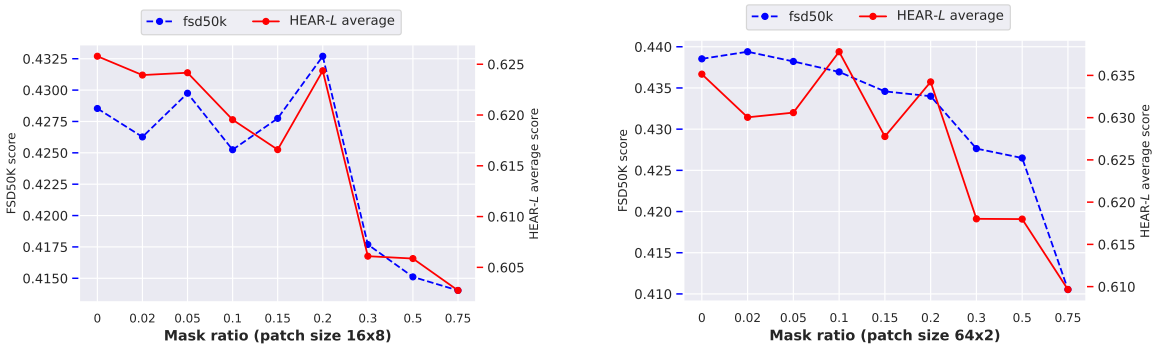
We consider several variations of the Audio Barlow Twins learning framework with the ViT / ViT$_C$ encoder architectures. Namely, **1)** we directly compare the performance of the ViT-*B* and ViT$_C$-*B* encoders, **2)** we consider the impact of using different frequency-time spectrogram patch sizes, and **3)** we consider introducing partial masking of one of the two spectrogram views during pre-training (inspired by MSN (Assran et al. (2022)) (Section 2.3.1.3)), such that the model is additionally tasked with performing "implicit denoising at the representation level" (Assran et al. (2022)). Identically as in Section 4.3, all ablation studies pre-train with Audio Barlow Twins for 100 epochs on the FSD50K development subset, using the same experimental set-up as used for full AudioSet pre-training (Section 4.1), except for the projector output dimension (set to 256), as well as AdamW *lr*/*wd* (we opt for our tuned *lr*/*wd* of $1 \cdot 10^{-4}/0.06$ (Section 4.1.1.1)).

4.4.1 ViT vs ViT$_C$

Figure 4.10a shows the training curves for the ViT-*B* and ViT$_C$-*B* models, both trained using a patch size of 16×16. Clearly, the ViT$_C$-*B* outperforms the ViT-*B* throughout training, supporting the claims made by Xiao et al. (2021) that the introduction of the convolutional stem significantly improves the performance of the ViT. This justifies why we only perform full AudioSet pre-training, which takes the best part of a week with a Transformer encoder, with the ViT$_C$ and not ViT models (as mentioned in Section 4.1).

4.4.2 Patch Size

We investigate the impact of the size of the patches extracted from the input spectrograms on the quality of the audio representations learned by the ViT$_C$-*B* encoder. We consider the patch sizes 16×16, 16×8 (default), 8×8, as well as splitting the spectrogram into time frames with 64×2 patches (where all inputs have 64 frequency bins). As shown in Table 3.2, whilst using a smaller patch size increases the capacity of the model (since there are more tokens which contribute to the multi-head self-attention mechanism), the number of flops, and subsequently the runtime, increases. From Figure 4.10b, we observe that reducing the patch size from 16×16 to 16×8, thereby doubling the number of patches as well as the patch time resolution, results in improved model performance. However, further reduction of the patch size to 8×8 results in minimal improvement to HEAR-*L* average score, in fact surprisingly worsening performance on FSD50K. This therefore justifies the use of 16×8 patches during full AudioSet pre-training, since using this patch size appears to optimally balance the trade-off between model capacity and model flops ($\sim$ half the flops as with 8×8 patches). We hypothesise that an explanation for this is that spectrograms appear to benefit from more local



(a) (b)

Figure 4.11: **Effect of partial view masking with different masking ratios**: We consider the effect of partial view masking of one of the two spectrogram views, inspired by recent work by Assran et al. (2022). We pre-train with different masking ratios, considering masking with both **(a)** 16×8 patches and **(b)** 64×2 patches. Results are shown both evaluated on FSD50K (blue) and the average score on the 5 HEAR-*L* tasks (red).

information processing, as demonstrated by the AudioNTT encoder, which encodes strong inductive biases, achieving far superior initial performance over the ViT$_C$ encoders. Therefore, increasing the number of patches through reduction of the patch size to 8×8, whilst increasing the model capacity, also increases the input sequence length, resulting in more patches being attended to within the multi-head self-attention mechanism. It is expected that with a significantly longer training schedule (we only train here for 100 epochs on FSD50K), where the model has sufficient time to learn this preference for local features, the performance with 8×8 patches will become superior. Further, we observe that the ViT$_C$ model appears to benefit from using frame-based 64×2 patches. However, we note that using a 64×2 patch size requires a convolutional stem with six 3×3 convolutions (Table 3.2), as opposed to only four with the other three patch sizes. Therefore, this imposes a stronger inductive bias of feature locality early in the stem of the ViT$_C$ encoder, which, as previously discussed, appears beneficial for the extraction of high-quality spectrogram features. As such, it is hard to disentangle this performance improvement in using frame-based patches from the increased inductive bias built-in to the model. Further, we anticipate that imposing this stronger inductive bias on the data limits the representational capacity of the model, and as such we choose not to use 64×2 patches during full AudioSet pre-training.

4.4.3 View Masking

Recall from Section 2.3.1.3 that MSN (Assran et al. (2022)) randomly drops a subset of the patches of one of the two image views, matching in feature space the masked view with the unmasked view, thereby performing "implicit denoising" (Assran et al. (2022)) at the representation level. MSN achieves SOTA results whilst simultaneously reducing computational and memory requirements, since the masked patches can be dropped before input into the ViT encoder. It is therefore of great interest to see whether adapting this approach to the audio domain can be beneficial in the pursuit of universal audio representations.

Similarly to MSN, we consider randomly masking patches from one of the two spectrogram views before input into the ViT$_C$ encoder. We implement random patch masking using the algorithm proposed by He et al. (2021), where the list of extracted patches is randomly shuffled, and the last M patches from the list are removed, where $M = rN$ (rounded to the nearest integer), with r being the masking ratio and N the initial number of patches. We consider partial view masking with both a patch size of 16×8 and 64×2, which both correspond to a total of $N = 48$ patches (with 64×96 spectrogram inputs).

Disappointingly, as shown in Figures 4.11a and 4.11b, partial view masking seems to harm the quality of the learned audio representations, with a clear trend that over a minimum threshold for r ($r \sim 0.2$,

corresponding to $M = 10$ masked patches), model performance is significantly reduced. Below this threshold we generally see slight degradation in model performance, although minimal variation (expected as only very few patches have been masked). We anticipate that masking a large number of spectrogram patches may fundamentally change the semantic content of the audio clip, such that matching the representations of the masked and unmasked views (through the invariance term of the Barlow Twins objective, Eqn3.5) encourages the model to embed together audio samples in representation space which no longer share the same semantic content, thereby damaging the quality of the learned representations. This is different to in CV, where strong masking doesn't visually appear to change the overall semantic content contained within an image (e.g. a heavily masked picture of a dog is still recognisable as a dog).

We additionally consider whether, instead of using a fixed masking ratio, slowly increasing the masking ratio during pre-training leads to improved representation quality. Starting the masking ratio at 0, we increase it to a value β at epoch 100 following a sinusoidal schedule with a warm up period of 10 epochs. However, initial experimentation, with $\beta = 0.3$, suggests that this also results in a degradation of model performance, although extensive analysis has not been performed.

Chapter 5

Conclusion and Future Work

This project aims to adapt the Barlow Twins learning framework to the audio domain, providing a new approach in the pursuit of universal audio representations. As such, we present ***Audio Barlow Twins***, a novel self-supervised audio representation learning method which does away with the necessity for negative samples or asymmetric learning updates. We evaluate the quality of the learned representations on 18 tasks from the HEAR Challenge. Our main contributions are as follows:

1. **Audio Barlow Twins pre-training produces high-quality audio representations**
   Audio Barlow Twins pre-training on AudioSet for 100 epochs with the AudioNTT encoder results in model performance which is on a par with, and in several cases better than, BYOL-A, as evaluated on the HEAR tasks. For example, the AudioNTT encoder achieves a test mAP of 0.474 on FSD50K, and an Onset FMS of 0.761 on DCASE 2016 Task 2. Both of these results are competitive with the SOTA for self-supervised audio representation learning. This demonstrates the robustness of the Audio Barlow Twins framework for the learning of general-purpose audio representations.

2. **The AudioNTT encoder is the strongest performer, but doesn't appear to benefit from longer training schedules**
   We consider several different encoder architectures: three convolutional encoders (AudioNTT, ResNet-18, and ResNet-18 ReGP NRF), as well as Transformer encoders, specifically the Vision Transformer (ViT) and the Vision Transformer with a convolutional stem (ViT$_C$, proposed by Xiao et al. (2021)). We demonstrate that whilst the AudioNTT encoder is the strongest performer, it doesn't benefit from a longer training schedule, rapidly optimizing in a few tens of epochs of pre-training on AudioSet. This is in contrast to the ViT$_C$ encoders, which continue to learn throughout the 100 epochs of full AudioSet pre-training. We anticipate that with a longer training schedule and more training data, the ViT$_C$-*B* will outperform the AudioNTT encoder. We additionally find that the ViT$_C$ models both outperform the vanilla ViTs and show less sensitivity to the optimization hyperparameters *lr* and *wd*, supporting claims made by Xiao et al. (2021).

3. **RRC benefits speech and environmental sound tasks but harms music tasks**
   Through thorough evaluation on the HEAR tasks, where we compare to randomly initialised baseline models, and extensive ablation studies, we discover a flaw with transferring instance-discrimination methods from CV to audio. Different audio downstream tasks benefit from different learned invariances, and as such benefit from different data augmentations which give rise to these invariances. For example, speech and environmental sound tasks appear to significantly benefit from invariance to slight pitch variations, as enforced by RRC. However, this invariance is hugely detrimental to performance on music tasks. As such, it is difficult to choose the optimal data augmentations that benefit all audio tasks universally. This provides motivation for using generative and predictive self-supervised approaches for audio representation learning, since they don't require the use of any data augmentations.

We envisage several research directions for future work that build on the findings from this project. Namely:

1. **Scaling-up ViT$_C$ Audio Barlow Twins pre-training**
   The ViT$_C$ encoders appear to benefit from long training schedules with large training datasets. We also observe a clear trend that ViT$_C$ performance scales with model size (ViT$_C$-*B* $>$ ViT$_C$-*S* $>$ ViT$_C$-*T*). It is of interest, therefore, to evaluate the impact on the quality of the learned audio representations through Audio Barlow Twins pre-training with larger ViT$_C$ models (e.g. ViT$_C$-*L* or ViT$_C$-*H*), training for more epochs on more data.

2. **Use of a CvT encoder**
   We observe that audio representations seem to benefit from feature locality, with the convolutional AudioNTT encoder optimizing rapidly as compared with the Transformer encoders, which are much slower to optimize. However, we hypothesise that the hard inductive biases of the AudioNTT encoder provides a ceiling to the quality of the learned audio representations, in contrast with the Transformer encoders, which have a much higher representational capacity. Hybrid convolutional-Transformer architectures, such as the Convolutional Vision Transformer (CvT) (Wu et al. (2021)), therefore present an attractive alternative, combining the preference for feature locality with the strong representational capacity and generalisation ability of the Transformer. In fact, Elbanna et al. (2022) found strong success in using the CvT within the BYOL-A learning framework, and as such we anticipate similar performance improvements using a CvT encoder within Audio Barlow Twins.

3. **Application of data augmentations directly on raw waveforms.**
   Unfortunately, due to an I/O bottleneck, we are unable to directly load in raw waveforms (.WAV files) during Audio Barlow Twins pre-training, instead pre-converting into mel-spectrograms (stored as .npy files). This limits the audio augmentations we can apply, and as such can only consider ones which act directly on spectrograms, whose effects are approximations to their desired effects on raw waveforms (e.g. RRC as an approximation to time stretch and pitch shift). If this I/O bottleneck can be overcome (such as possibly through storing the waveforms as HDF5 files), then it is of great interest to consider the effect on different downstream tasks of different augmentations that act directly on raw waveforms, within the Audio Barlow Twins learning framework. Applying the augmentations directly on raw waveforms, instead of spectrograms, allows for a) a better control of the strength of these augmentations (as it is possible to *listen* directly to their effect), and b) a greater number of augmentations to be considered (e.g. pitch shift, time masking, time shift, time stretch, fade in/out, compression, etc.).

Bibliography

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *CoRR*, abs/1907.10902. pages 34

Al-Tahan, H. and Mohsenzadeh, Y. (2020). Clar: Contrastive learning of auditory representations. pages 14, 19, 20

Anantapadmanabhan, A., Bellur, A., and Murthy, H. A. (2014). Mridangam stroke dataset. pages 31

Assran, M., Caron, M., Misra, I., Bojanowski, P., Bordes, F., Vincent, P., Joulin, A., Rabbat, M., and Ballas, N. (2022). Masked siamese networks for label-efficient learning. pages 1, 2, 4, 15, 17, 18, 27, 48, 50, 51

Baevski, A., Zhou, H., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *CoRR*, abs/2006.11477. pages 1, 7, 19, 39

Barlow, H. B. and Rosenblith, W. A. (1961). *Possible principles underlying the transformations of sensory messages*, pages 217–234. MIT Press. pages 5

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. pages 7

Cao, H., Cooper, D. G., Keutmann, M. K., Gur, R. C., Nenkova, A., and Verma, R. (2014). Crema-d: Crowd-sourced emotional multimodal actors dataset. *IEEE Transactions on Affective Computing*, 5(4):377–390. pages 31

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. *CoRR*, abs/2005.12872. pages 7

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. (2018). Deep clustering for unsupervised learning of visual features. *CoRR*, abs/1807.05520. pages 15

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *CoRR*, abs/2006.09882. pages 15, 17

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294. pages 1, 16, 17, 21, 27, 48

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. (2020). A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709. pages 1, 4, 13, 15, 19, 46

Chen, X. and He, K. (2020). Exploring simple siamese representation learning. *CoRR*, abs/2011.10566. pages 4

Chen, X., Xie, S., and He, K. (2021). An empirical study of training self-supervised vision transformers. *CoRR*, abs/2104.02057. pages 1, 10, 27, 43

Cooper, S. and Shaw, S. (2020). Gunshots recorded in an open field using ipod touch devices. pages 31

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. pages 1, 4

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805. pages 1, 7, 18, 27

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929. pages 2, 7, 9, 10, 16, 26, 27, 43

Elbanna, G., Scheidwasser-Clow, N., Kegler, M., Beckmann, P., Hajal, K. E., and Cernak, M. (2022). Byol-s: Learning self-supervised speech representations by bootstrapping. pages 31, 39, 40, 54

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., and Norouzi, M. (2017). Neural audio synthesis of musical notes with wavenet autoencoders. pages 31

Ericsson, L., Gouk, H., and Hospedales, T. M. (2020). How well do self-supervised models transfer? *CoRR*, abs/2011.13377. pages 1, 3

Ericsson, L., Gouk, H., Loy, C. C., and Hospedales, T. M. (2021). Self-supervised representation learning: Introduction, advances and challenges. *CoRR*, abs/2110.09327. pages 4, 12

Fonseca, E., Favory, X., Pons, J., Font, F., and Serra, X. (2020a). Fsd50k. pages 31

Fonseca, E., Ortego, D., McGuinness, K., O'Connor, N. E., and Serra, X. (2020b). Unsupervised contrastive learning of sound event representations. pages 5, 19, 20, 23, 24

Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., and Ritter, M. (2017). Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. pages 2, 4, 30

Gong, Y., Chung, Y., and Glass, J. R. (2021a). PSLA: improving audio event classification with pretraining, sampling, labeling, and aggregation. *CoRR*, abs/2102.01243. pages 4, 30

Gong, Y., Lai, C. J., Chung, Y., and Glass, J. R. (2021b). SSAST: self-supervised audio spectrogram transformer. *CoRR*, abs/2110.09784. pages 1, 30, 41

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. (2005). Measuring statistical dependence with hilbert-schmidt norms. volume 3734. pages 47

Grill, J., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. Á., 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. *CoRR*, abs/2006.07733. pages 1, 4, 15, 16, 20, 21, 47, 48

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C. A., Dieleman, S., Elsen, E., Engel, J. H., and Eck, D. (2018). Enabling factorized piano music modeling and generation with the MAESTRO dataset. *CoRR*, abs/1810.12247. pages 31

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2021). Masked autoencoders are scalable vision learners. pages 1, 18, 51

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385. pages 6, 7, 15, 26, 27

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., and Li, M. (2018). Bag of tricks for image classification with convolutional neural networks. *CoRR*, abs/1812.01187. pages 9, 27

Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J., and Wilson, K. W. (2016). CNN architectures for large-scale audio classification. *CoRR*, abs/1609.09430. pages 6

Hsu, W., Bolte, B., Tsai, Y. H., Lakhotia, K., Salakhutdinov, R., and Mohamed, A. (2021). Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *CoRR*, abs/2106.07447. pages 1, 7, 19

Huang, P.-Y., Xu, H., Li, J., Baevski, A., Auli, M., Galuba, W., Metze, F., and Feichtenhofer, C. (2022). Masked autoencoders that listen. pages 1, 30, 41

Hutter, F., Hoos, H., and Leyton-Brown, K. (2014). An efficient approach for assessing hyperparameter importance. *31st International Conference on Machine Learning, ICML 2014*, 2:1130–1144. pages 35

Khan, S. H., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., and Shah, M. (2021). Transformers in vision: A survey. *CoRR*, abs/2101.01169. pages 7

Kim, B. and Pardo, B. (2018). Vocal Imitation Set v1.1.3 : Thousands of vocal imitations of hundreds of sounds from the AudioSet ontology. pages 31

Kim, J. W., Salamon, J., Li, P., and Bello, J. P. (2018). Crepe: A convolutional representation for pitch estimation. pages 39

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. pages 31

Koizumi, Y., Takeuchi, D., Ohishi, Y., Harada, N., and Kashino, K. (2020). The ntt dcase2020 challenge task 6 system: Automated audio captioning with keywords and sentence length estimation. pages 20, 26

Koutini, K., Eghbal-zadeh, H., and Widmer, G. (2021). Receptive field regularization techniques for audio classification and tagging with deep convolutional neural networks. *CoRR*, abs/2105.12395. pages 26, 27

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc. pages 6

Lacoste, A., Luccioni, A., Schmidt, V., and Dandres, T. (2019). Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*. pages 60

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. pages 6

Lecun, Y. and Misra, I. (2021). Self-supervised learning: The dark matter of intelligence. pages 12

Liu, A. T., wen Yang, S., Chi, P.-H., chun Hsu, P., and yi Lee, H. (2020). Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. pages 12, 28

Liu, S., Mallol-Ragolta, A., Parada-Cabeleiro, E., Qian, K., Jing, X., Kathan, A., Hu, B., and Schuller, B. W. (2022). Audio self-supervised learning: A survey. pages 19

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692. pages 1, 7

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. pages 9

Loshchilov, I. and Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. pages 33

Loshchilov, I. and Hutter, F. (2017). Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101. pages 10

McFee, B., Raffel, C., Liang, D., Ellis, D., Mcvicar, M., Battenberg, E., and Nieto, O. (2015). librosa: Audio and music signal analysis in python. pages 18–24. pages 31

Mesaros, A., Heittola, T., Benetos, E., Foster, P., Lagrange, M., Virtanen, T., and Plumbley, M. D. (2018). Detection and classification of acoustic scenes and events: Outcome of the DCASE 2016 challenge. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(2):379–393. pages 31

Niizumi, D., Takeuchi, D., Ohishi, Y., Harada, N., and Kashino, K. (2021). Byol for audio: Self-supervised learning for general-purpose audio representation. pages 1, 5, 20, 21, 23, 24, 25, 26, 30, 33, 38, 39, 44

Niizumi, D., Takeuchi, D., Ohishi, Y., Harada, N., and Kashino, K. (2022a). Byol for audio: Exploring pre-trained general-purpose audio representations. pages 20, 21, 23, 24, 25, 26, 27, 37, 38, 45

Niizumi, D., Takeuchi, D., Ohishi, Y., Harada, N., and Kashino, K. (2022b). Masked spectrogram modeling using masked autoencoders for learning general-purpose audio representation. pages 1, 30, 41

Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. *CoRR*, abs/1604.07379. pages 1, 12

Piczak, K. J. (2015). ESC: Dataset for Environmental Sound Classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pages 1015–1018. ACM Press. pages 31

Radosavovic, I., Kosaraju, R. P., Girshick, R. B., He, K., and Dollár, P. (2020). Designing network design spaces. *CoRR*, abs/2003.13678. pages 9

Rasley, J., Rajbhandari, S., Ruwase, O., and He, Y. (2020). Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '20, page 3505–3506, New York, NY, USA. Association for Computing Machinery. pages 26, 28

Saeed, A., Grangier, D., and Zeghidour, N. (2020). Contrastive learning of general-purpose audio representations. *CoRR*, abs/2010.10915. pages 1, 5, 19, 20, 30

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. pages 6

Stöter, F.-R., Chakrabarty, S., Habets, E., and Edler, B. (2018). Libricount, a dataset for speaker count estimation. pages 31

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 843–852. pages 4

Tian, M., Srinivasamurthy, A., Sandler, M., and Serra, X. (2014). Beijing opera percussion instrument dataset. pages 31

Tian, Y., Chen, X., and Ganguli, S. (2021). Understanding self-supervised learning dynamics without contrastive pairs. *CoRR*, abs/2102.06810. pages 4, 16

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2020). Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877. pages 7, 9, 10

Tsai, Y.-H. H., Bai, S., Morency, L.-P., and Salakhutdinov, R. (2021). A note on connecting barlow twins with negative-sample-free contrastive learning. pages 22, 47

Turian, J., Shier, J., Khan, H. R., Raj, B., Schuller, B. W., Steinmetz, C. J., Malloy, C., Tzanetakis, G., Velarde, G., McNally, K., Henry, M., Pinto, N., Noufi, C., Clough, C., Herremans, D., Fonseca, E., Engel, J., Salamon, J., Esling, P., Manocha, P., Watanabe, S., Jin, Z., and Bisk, Y. (2022). Hear: Holistic evaluation of audio representations. pages 2, 21, 30, 31

Tzanetakis, G. and Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302. pages 31

Valk, J. and Alumäe, T. (2020). Voxlingua107: a dataset for spoken language recognition. pages 31

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. pages 1, 7, 27

Warden, P. (2018). Speech commands: A dataset for limited-vocabulary speech recognition. pages 31

Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., and Zhang, L. (2021). Cvt: Introducing convolutions to vision transformers. pages 54

Xiao, T., Singh, M., Mintun, E., Darrell, T., Dollár, P., and Girshick, R. B. (2021). Early convolutions help transformers see better. *CoRR*, abs/2106.14881. pages 10, 11, 26, 28, 34, 36, 37, 50, 53

You, Y., Gitman, I., and Ginsburg, B. (2017). Large batch training of convolutional networks. pages 33, 35

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. *CoRR*, abs/2103.03230. pages 2, 4, 5, 21, 22, 33, 34, 36, 40, 46, 47

Zhao, N., Wu, Z., Lau, R. W. H., and Lin, S. (2020). What makes instance discrimination good for transfer learning? pages 3

Zhou, J., Wei, C., Wang, H., Shen, W., Xie, C., Yuille, A. L., and Kong, T. (2021). ibot: Image BERT pre-training with online tokenizer. *CoRR*, abs/2111.07832. pages 1, 4, 27

Appendix A

Legal, Social, Ethical and Professional Considerations

A.1 Datasets

We use a total of 16 datasets for this project, AudioSet and 15 HEAR datasets.

1. **AudioSet**: AudioSet is available under the Creative Commons Attribution 4.0 International (CC BY 4.0) license, while the AudioSet ontology is available under a Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) license. Covered under these licenses is the sharing, copying and re-distributing of AudioSet for non-commercial research purposes.
2. **HEAR**: The HEAR datasets are available for download from `https://zenodo.org/record/5887964`, which is covered by a Creative Commons Attribution 4.0 International License. This license allows for use of these datasets for non-commercial research purposes.

A.2 Environmental Impact

Experiments are conducted using private infrastructure, specifically the Imperial High Performance Computing Cluster, which has an estimated carbon efficiency of 0.432 kgCO_2eq/kWh (European average). A cumulative total of ~ 2000 hours of computation has been performed on NVIDIA Quadro RTX A6000 GPUs (TDP of 300W). Total emissions are estimated to be 259.2 kgCO_2eq. These estimations have been conducted using the Machine Learning Impact calculator (Lacoste et al. (2019)). This is approximately equivalent to the CO_2 emssisions of a single economy-class flight to Rome (220 kgCO_2eq[1]).

[1] Calculated using `https://calculator.carbonfootprint.com/calculator.aspx?lang=en-GB&tab=3`.