SAPIENZA
UNIVERSITÀ DI ROMA

Joint Embedding Transformer for Self-Supervised Audio Representation

Faculty of Information Engineering, Informatics, and Statistics
Master of Science in Engineering in Computer Science

Riccardo Salvalaggio
ID number 1750157

Advisor
Prof. Danilo Comminiello

Academic Year 2021/2022

Thesis defended on 31/01/2023
in front of a Board of Examiners composed by:

Proff.: Danilo Comminiello,Giorgio Grisetti, Christian Napoli, Fabrizio Silvestri, Andrea Vitaletti.
Prof. Aristidis Anagnostopoulos (chairman)

Joint Embedding Transformer for Self-Supervised Audio Representation
Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: salvalaggio.1750157@studenti.uniroma1.it

Dedicated to
my Grandmother, my Mum,
my Brother, and my Girlfriend
who always inspired me.
I hope I'm making you proud.

Abstract

Self-supervised learning has emerged as a promising approach for training machine learning models on large datasets of unlabeled data. One area where self-supervised learning has shown particular promise is in the processing and classification of audio data. Audio Spectrogram Transformer (AST) is a machine learning model that has been developed to process and classify audio data using a self-supervised learning approach.
Despite the impressive performance of AST on a number of audio classification tasks, there is still room for improvement in its ability to process and classify audio data accurately and efficiently. In this thesis, we propose a potential improvement to AST that could enhance its performance on audio classification tasks and increase its robustness and scalability.
The proposed improvements follow the AST implementations using Self-Supervised learning for more fast and more efficient training of AST. We explore the potential of AST to learn from large amounts of unlabeled data and to handle noise and other distortions in the audio data.
The chosen solution is VCR-AST: a Self-Supervised Learning Framework using Variance-Invariance-Covariance Regularization. The results of our experiments show that the proposed method for AST is promising and potentially comparable to the original SSAST implementation. This work demonstrates the potential of self-supervised learning approaches to be used in a wide range of audio processing tasks and applications and provides insights into the potential of these approaches to improve the performance of machine learning models on large datasets of unlabeled data.

Keywords: Deep Learning, Transformers, VICReg, Self-Supervised Learning, AST, Audio tasks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Audio Spectrogram Transformer (AST)[1] is a machine learning model that has been developed to process and classify audio data. It is based on the Transformer[2] architecture, which is a type of neural network that has been widely used in natural language processing tasks. AST has been shown to be particularly effective at processing and classifying audio data using a self-supervised learning approach, which allows it to learn useful features and representations of the data without the need for explicit labels.
Despite the impressive performance of AST on a number of audio classification tasks, there is still room for improvement in its ability to process and classify audio data accurately and efficiently. In this thesis, we propose a Self-Supervised method applied to AST that could enhance its performance on audio classification tasks and increase its robustness and scalability.
We also explore the potential of AST to learn from large amounts of unlabeled data and to handle noise and other distortions in the audio data.
The goals of this thesis are to demonstrate the potential of these improvements to enhance the performance of AST on audio classification tasks, and to provide insights into the potential of AST and other machine learning models to be used in a wider range of audio processing tasks and applications.
Overall, this thesis aims at contribute to the understanding of the capabilities and limitations of Audio Spectrogram Transformer (AST) as a tool for processing and classifying audio data, and to identify potential areas for further research and development that could enhance the effectiveness and efficiency of AST .
The remainder of this thesis is organized as follows. In Section 2, we provide a deep discussion of AST and its building blocks, In section 3, we provide a review of related work on machine learning approaches for audio processing and classification, including the development and performance of AST and the application of Self-Supervised Learning Framework. In Section 4, we describe the proposed improvements to AST, VCR [3], its implementation, and promised improvements, and also, the news of this thesis. In Section 5, we present the results of our experiments on a number of audio classification tasks and Datasets, evaluating the performance of SSAST with and without the proposed improvements. Also, provide an analysis of the potential impact and limitations of the proposed improvements to AST. Finally, in Section 6, we summarize our findings and conclude with a discussion of future work.

Chapter 2

Audio Spectogram Transformer

AST, Audio Spectrogram Transformer, is a model that uses attention mechanisms and no convolutions for audio classification. This allows it to capture long-range global context early on and to be applied directly to spectrograms.
Transfer learning from ViT [4], a model trained on ImageNet [5], has been shown to significantly improve AST's performance.[6][7][8]
In comparison to other audio classification methods, AST performs better on a variety of tasks and datasets, including AudioSet[9], ESC-50[10], and Speech Commands[11].
It can process variable-length inputs, can be used for multiple tasks without changing its architecture, has a simpler architecture with fewer parameters, and has faster training convergence compared to state-of-the-art CNN-attention hybrid models.
AST is thus a promising approach for audio classification that offers improved performance and flexibility.
Its ability to handle variable-length inputs is especially useful because audio recordings can vary greatly in length.
Fixed-length input requirements can result in lost information for short inputs or inefficient processing for long inputs, but AST can process the audio of any length without loss of information or efficiency.
This makes it suitable for a range of audio classification tasks, including audio event classification, speech command recognition, and emotion recognition.
AST's simplicity is a major advantage because complex models with many layers

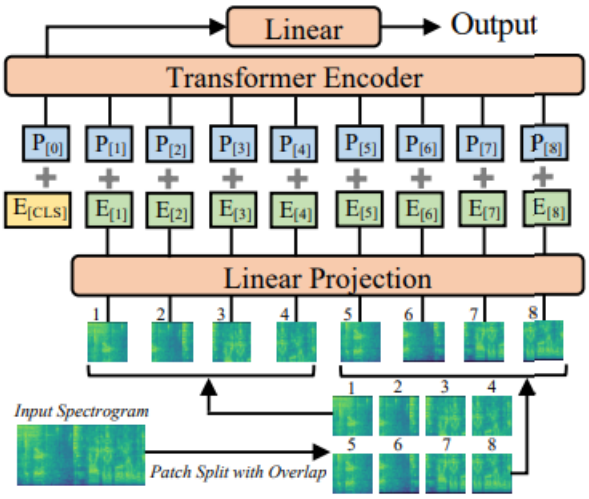


Figure 2.1. The proposed AST architecture involves dividing a 2D audio spectrogram into a series of overlapping 16x16 patches. These patches are then transformed into a sequence of 1D patch embeddings through a linear projection. A learnable positional embedding is added to each patch embedding, and a classification token is inserted at the beginning of the sequence. The resulting sequence of embeddings is processed through a Transformer, and the output of the classification token is used to classify the input spectrogram with a linear layer.

and parameters can be challenging to train and prone to overfitting.
AST's architecture is relatively simple, requiring fewer parameters, making it easier to train and less prone to overfitting.

Additionally, AST has faster convergence during training compared to CNN-attention hybrid models, which means it can achieve good performance in fewer training iterations. The simplicity of AST's architecture is beneficial in terms of both training efficiency and generalizability.
AST can be easily fine-tuned for different audio classification tasks, unlike traditional audio classification models that often require task-specific architecture modifications or feature engineering. This makes AST a versatile and efficient tool for audio classification. AST's attention-based architecture allows it to highlight the most important parts of the input for making a classification decision, improving the interpretability of audio classification models and providing insight into the underlying process.
Overall, the Audio Spectrogram Transformer (AST) is a promising approach for audio classification that offers improved performance, flexibility, and interpretability compared to traditional methods.

2.1 Building Blocks

2.1.1 Patch Embedding Layer

Patch embedding is a technique that takes inspiration from the BERT architecture [12] by tokenizing input sequences, i.e. by prepending class information to a sequence of embedded patches. It is used to flatten each 16x16 patch into a 1D patch embedding of size 768 using a linear projection layer. The projection layer

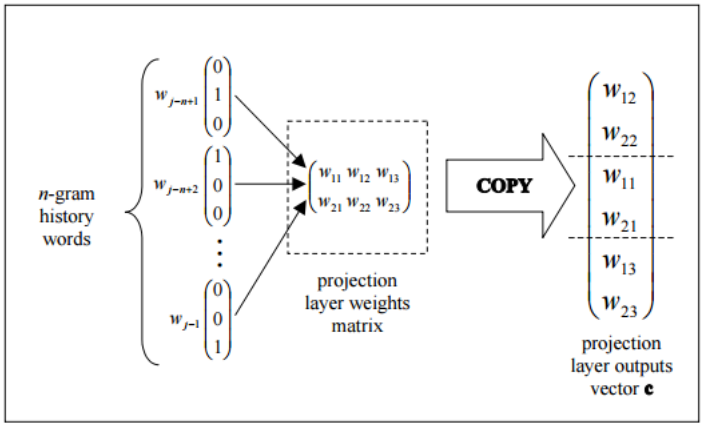


Figure 2.2. LinearProjection layer

transforms discrete indices representing an n-gram context into a continuous vector space using shared weights. This allows for multiple instances of the same word in a context to be processed using the same weights, effectively increasing the amount of training data for the projection layer. This process can be thought of as a single convolution layer with a large kernel and stride size, similar to a 1x1 convolution in each Transformer block.

2.1.2 Positional Encoding

To capture the spatial structure of the 2D audio spectrogram, a trainable positional embedding is added to each patch embedding in the Transformer architecture. This positional embedding has a size of 768 and allows the model to make use of the patch sequence's order, even though the Transformer architecture does not capture input order information and the patch sequence is not in the temporal order. The positional encodings, which have the same dimension as the embeddings and are usually summed with them, are commonly computed as follows:

$$\begin{aligned} PE_{pos,2i} &= \sin \frac{pos}{10000^{\frac{2i}{d_{model}}}} \\ PE_{pos,2i+1} &= \cos \frac{pos}{10000^{\frac{2i}{d_{model}}}} \end{aligned} \tag{2.1}$$

where pos is the position and i is the dimension. In other words, each dimension of the positional encoding corresponds to a sinusoid.

2.1.3 Transformer

The proposed Audio Spectrogram Transformer is based on the Transformer architecture [2], which has gained significant attention in the field of deep learning. Classical sequence transduction models often utilize complex recurrent or convolutional neural networks that include an encoder and a decoder, with the best-performing models also incorporating an attention mechanism between the encoder and decoder. The Transformer aims to simplify this architecture by replacing recurrence and convolutions with attention mechanisms alone. In this model, the encoder maps an input sequence of symbol representations to a sequence of continuous representations, and at each step the model is autoregressive, using previously generated symbols as additional input when generating the next symbol. The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder.

2.1.4 Attention

One of the key features of this architecture is attention, which is a mechanism developed to improve the performance of encoder-decoder architectures on neural network-based machine translation tasks. Attention allows the decoder to access information from every encoder's hidden state through an interface that connects the encoder and decoder. This enables the model to selectively focus on valuable parts of the input sequence and learn the associations between them, which helps the model effectively handle long input sentences. The process of attention involves the following steps:

1. The encoder LSTM[13] processes the entire input sentence and encodes it into a context vector, which is the final hidden state of the LSTM/RNN. This context vector is intended to be a good summary of the input sentence. The intermediate states of the encoder are ignored, and the final state serves as the initial hidden state of the decoder.

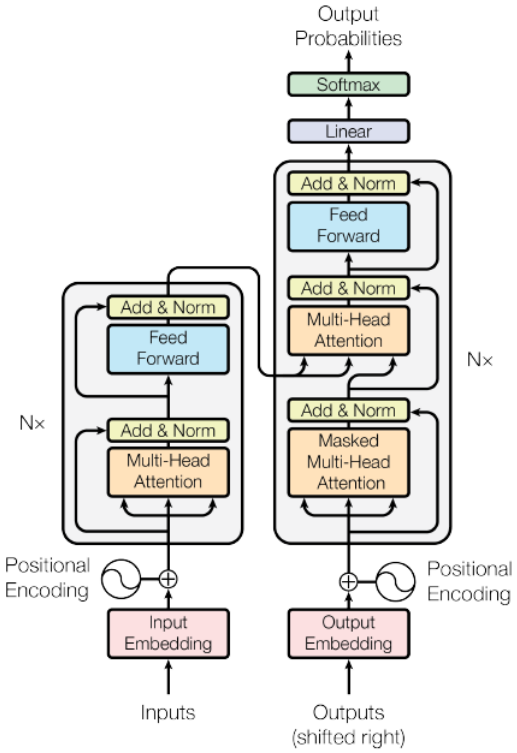


Figure 2.3. Transformer Architecture

2. The decoder LSTM or RNN units produce the words in a sentence one after another.

Finally, the Transformer encoder's output of the [CLS] token serves as the audio spectrogram representation. A linear layer with sigmoid activation maps this representation to labels for classification.

2.1.5 Audio Spectrograms as Images

As previously mentioned, most of the applications of Transformer involve image tasks or NLP models because Transformer requires more data to train than CNNs and only outperforms CNN when the data exceeds 14 million samples. Another criticism of Transformer is that it is difficult to achieve good results on audio tasks because there are no audio datasets of comparable size. The only way to train the model effectively is to use cross-modality transfer learning from another model.

Image-Audio Transfer learning

Although transfer learning is not a new concept, it is unusual to apply it from a model pre-trained for image tasks to audio. However, in this case, transfer learning was applied to an off-the-shelf Vision Transformer, rather than a CNN model, making it a novel application. Transfer learning is a technique in machine learning where a model trained on one task is re-purposed for a related task. It is often used in deep

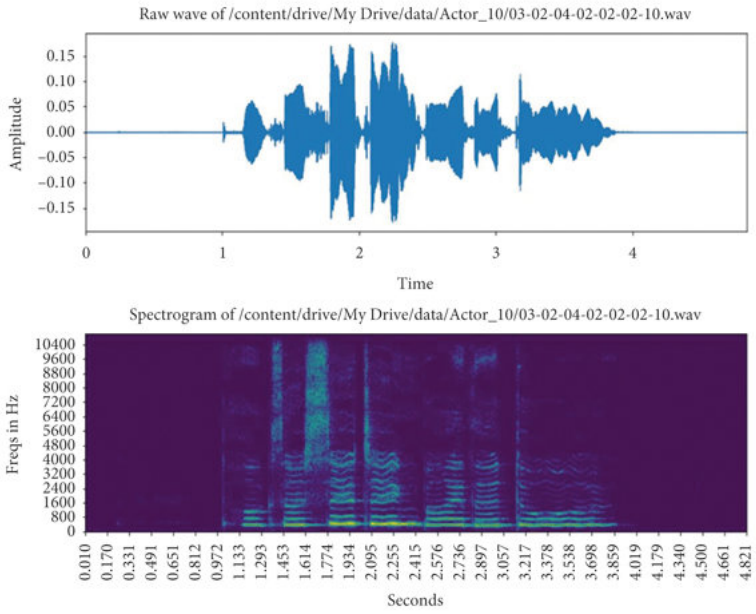


Figure 2.4. Audio waveform to Spectogram

learning due to the large amount of resources required to train deep learning models or the size and complexity of the datasets on which they are trained. The idea behind transfer learning is that if a model is trained on a large and diverse enough dataset, it can act as a generic model that can be fine-tuned for a specific task. For example, knowledge gained from learning to recognize cars could be applied to the problem of recognizing trucks. In practice, this involves using a pre-trained Vision Transformer model trained on the ImageNet dataset and retraining it on an audio dataset (such as ESC-50, SpeechCommands, or AudioSet).

Vision Transformer

In this case, as previously mentioned, transfer learning has been applied to the Vision Transformer (ViT) [4]. While ViT and the Audio Spectrogram Transformer (AST) have similar architectures (e.g., both use a standard Transformer and have the same patch size and embedding size), they are not identical. Therefore, some modifications are needed for the adaptation. First, the input to ViT is a 3-channel image while the input to AST is a single-channel spectrogram. To adapt the patch embedding layer of ViT to AST, the authors average the weights corresponding to each of the three input channels of the ViT patch embedding layer and use these averaged weights as the weights of the AST patch embedding layer. In addition, the length of an audio spectrogram can be variable with an upper bound. While the Transformer naturally supports variable input length and can be directly transferred from ViT to AST, the positional embedding must be carefully processed because it learns to encode spatial information during ImageNet training. Finally, since the classification task is different, the last classification layer is removed and a new one is initialized for AST.

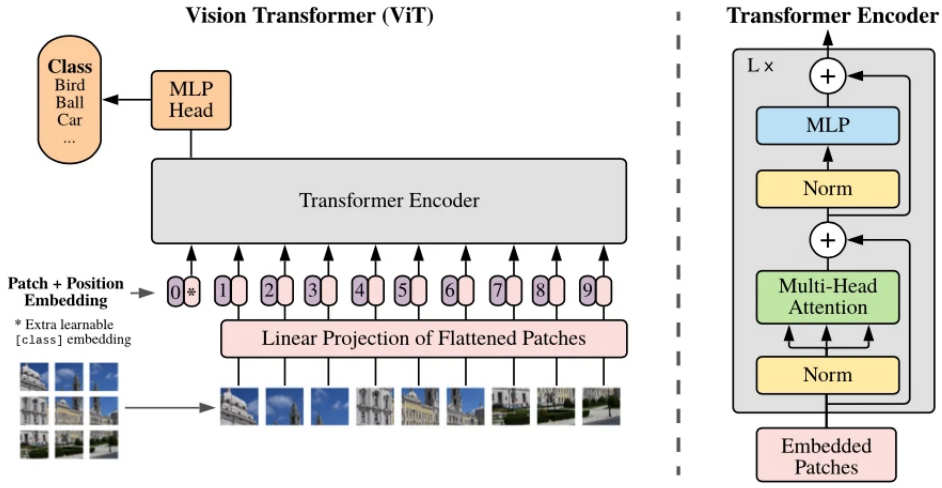


Figure 2.5. ViT Architecture

An image is divided into small patches in this approach, and each patch may contain 16x16 pixels. The input sequence consists of a flattened (1D) vector of pixel values from a patch of size 16x16. Each flattened element is fed into a linear projection layer that produces the "Patch embedding." Positional embeddings are then linearly added to the sequence of image patches so that the images retain their positional information. These embeddings inject information about the relative or absolute position of the image patches in the sequence. A learnable class embedding is attached to the sequence according to the position of the image patch, and this class embedding is used to predict the class of the input image after being updated by self-attention. Classification is performed by stacking an MLP head on top of the Transformer at the position of the learnable embedding added to the sequence.

2.1.6 Method

The training pipeline for the Audio Spectrogram Transformer (AST) involves several steps:

- **Preprocessing:** The raw audio data is typically transformed into a spectrogram representation, which captures the spectral content of the audio over time. This can be done using techniques such as windowing and Fast Fourier Transform (FFT). The window size, overlap, and FFT size can all be hyperparameters that can be tuned to achieve the desired trade-off between frequency resolution and time resolution in the spectrogram.

- **Data preparation:** Once the spectrograms have been generated, they are typically split into overlapping patches and the patches are transformed into a sequence of patch embeddings using a linear projection layer. The patch size and the overlapping can be hyperparameters that can be tuned to achieve the desired balance between the local context and global context in the patches. The patch embeddings are then augmented with positional embeddings, which

encode the spatial information of the patches within the spectrogram. These embeddings can be learned during training or precomputed based on the patch position.

- **Model training:** The AST model, which consists of a standard Transformer architecture with an MLP head for classification, is trained on the patch sequences using a supervised learning approach. The model is optimized using an appropriate loss function (e.g., cross-entropy loss for classification tasks) and a gradient-based optimization algorithm (e.g., stochastic gradient descent). The learning rate, batch size, and the number of training epochs can all be hyperparameters that can be tuned to achieve the desired convergence rate and generalization performance of the model. The model may also be trained using techniques such as data augmentation, regularization, and early stopping to improve its generalization ability.

- **Evaluation:** Once the model has been trained, it is typically evaluated on a held-out test set to assess its performance on the target task. This may involve computing evaluation metrics such as accuracy, F1 score, precision, recall, or AUC. The model may also be evaluated using visualization techniques such as confusion matrices or learning curves.

- **Fine-tuning:** If necessary, the AST model can be fine-tuned on a specific task or dataset by continuing to train the model using labeled data for that task. This can help to improve the performance of the model on the specific task or dataset. Fine-tuning may involve adjusting the model architecture, hyperparameters, or training methodology to better suit the characteristics of the specific task or dataset.

2.2 Audio Tasks

Typical Tasks for Neural Networks in the Audio scene are:

- **Speech recognition**: Neural networks can be used to recognize and transcribe spoken language from audio data. This is a challenging task, as it requires the model to be able to understand and interpret spoken language, as well as to recognize different accents and languages.

- **Audio classification**: Neural networks can be used to classify audio data based on certain characteristics, such as the type of sound (e.g. speech, music, noise), the speaker, or the language being spoken.

- **Music generation**: Neural networks can be used to generate music by learning patterns and features in existing music and using this knowledge to generate new music that is similar to the training data.

AST provides applications for most of them and in addition, has shown good potential in audio scene analysis and audio localization.

2.2.1 Contribution

Recently, the Transformer has also been adapted for audio processing but is typically used in conjunction with a CNN. Other efforts combine CNNs with simpler attention modules. The proposed AST differs from these studies in that it is convolution-free and purely based on attention mechanisms. The closest work is the Vision Transformer (Vit), which is a Transformer architecture for vision tasks. AST and ViT have similar architectures but it has only been applied to fixed-dimensional inputs (images) while AST can process variable-length audio inputs. In addition, this work has proposed an approach to transfer knowledge from ImageNet pre-trained ViT to AST.

One of the key contributions of AST is its ability to learn from a wide range of audio data, including both speech and non-speech sounds. This has made it a valuable tool for tasks such as speech recognition, where the model can learn from a diverse range of audio data and potentially improve its performance. Additionally, AST has been shown to be effective at learning from a wide range of languages and accents, which makes it a useful tool for speech recognition tasks in a variety of languages.

2.2.2 Boosting AST's performance

Some possible areas of improvement for AST might include:

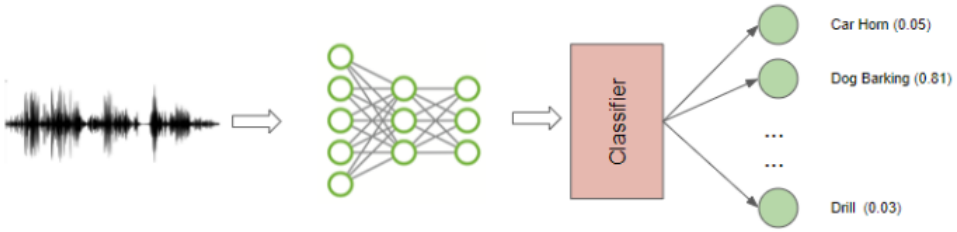
- **Increased robustness to noise and other distortions**: AST and other machine learning models for audio processing can be sensitive to noise and other distortions in the audio data. Future improvements to AST could focus on increasing its robustness to these types of distortions, which would make it more useful in real-world applications.
- **Enhanced ability to learn from large amounts of unlabeled data:** Self-supervised learning approaches like AST have the advantage of being able to learn from large amounts of unlabeled data. Future improvements to AST could focus on increasing its ability to learn from these types of datasets and make use of more of the available data for training.
- **Improved efficiency**[14] **and scalability:** Future improvements to AST could focus on increasing the efficiency and scalability of the model, which would allow it to handle larger and more complex datasets and enable it to be used in more resource-constrained environments.

Overall, it is likely that future improvements to the Audio Spectrogram transformer (AST) will focus on increasing its effectiveness and efficiency at processing and classifying audio data, as well as on making it more robust and scalable. These improvements could enable AST and other machine learning models to be used in a wider range of audio processing tasks and applications.

Chapter 3

Self-Supervised Learning for AST

Audio data in day-to-day life can come in innumerable forms such as human speech, music, animal voices, and other natural sounds as well as man-made sounds from human activity such as cars and machinery. Given the prevalence of sounds in our lives and the range of sound types, it is not surprising that there are a vast number of usage scenarios that require us to process and analyze audio. Now that deep learning has come of age, it can be applied to solve a number of use cases. Obviously, the possible applications are vast. This could be applied to detect the



failure of machinery or equipment based on the sound that it produces, or in a surveillance system, to detect security break-ins.

3.1 Audio Self-Supervised Learning

Inspired by the humans' cognitive ability to generalize knowledge and skills, Self-Supervised Learning (SSL) targets at discovering general representations from large-scale data without requiring human annotations, which is an expensive and time-consuming task. Its success in the fields of computer vision and natural language processing has prompted its recent adoption into the field of audio and speech processing. Comprehensive reviews summarising the knowledge in audio SSL are currently missing. To fill this gap, in the present work, we provide an overview of the SSL methods used for audio and speech processing applications. Herein, we also summarise the empirical works that exploit the audio modality in multimodal SSL frameworks, and the existing suitable benchmarks to evaluate the power of SSL in the computer audition domain. Finally, we discuss some open problems and point out the future directions on the development of audio SSL. Exactly like children, during the process of learning in order to understand the world, the so-called 'schemas', i.e., higher-order cognitive structures that have been hypothesized to underlie many aspects of human knowledge and skill emerge. **According to Piaget, children's development is interpreted through an equilibration mechanism that explains how new information is balanced according to old knowledge.** Inspired by the cognitive process of developing dynamic structures with the capability of generalization, the Self-Supervised Learning (SSL) paradigm has been presented. SSL[15] mitigates two difficulties that currently limit the application of deep learning: the need for human annotations and the difficulty in designing effective network architectures for specific tasks. First, the current success of deep learning reckons on big data which typically consumes uninhibited human efforts in annotations. This faces the issue of annotation bias as well as

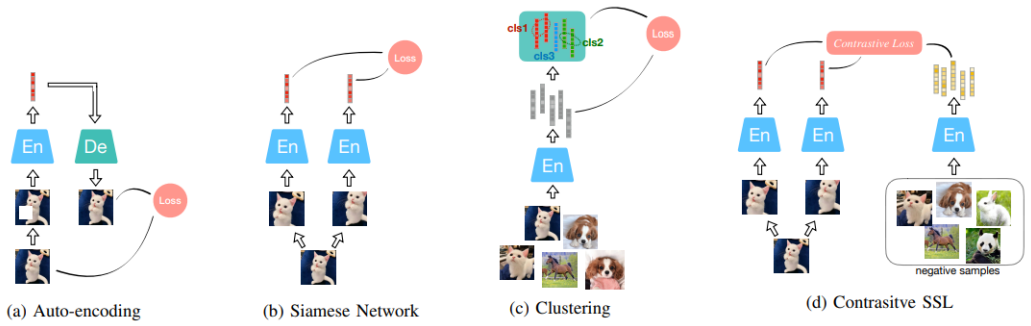


Figure 3.1. There are four different frameworks for predictive self-supervised learning: (a) auto-encoding, (b) Siamese networks, (c) clustering, and (d) contrastive learning. The auto-encoding framework involves an encoder and a decoder, where the encoder learns representations from distorted input signals and the decoder attempts to recover the clean signal from those representations. In the Siamese network framework, two views of the same data point are processed, and the latent representation of one sub-network is used as a pseudo-label for the other sub-network. Clustering is applied in the clustering framework to group learned representations, with the clustering centroids serving as pseudo-labels for training. Finally, the contrastive learning framework constructs a contrastive loss through the use of negative samples. All of these frameworks use pseudo-labels to construct training objectives.

the fact that annotation procedures often cannot optimally preserve data privacy. Second, as long as the pretext model can generate proper representations of the data, these can be used for multiple downstream tasks, reducing, at the same time, the difficulty in designing reliable downstream models.

3.2 Approaches

Self-supervised representation learning has made significant progress over the last years, almost reaching the performance of supervised baselines on many downstream tasks. Several recent approaches rely on a **joint embedding architecture**[16] in which two networks are trained to produce similar embeddings for different views of the same image. The main challenge with joint embedding architectures is to prevent a collapse in which the two branches ignore the inputs and produce identical and constant output vectors. There are two main approaches to preventing collapse: contrastive methods and information maximization methods. **Contrastive methods** [17] tend to be costly, require large batch sizes or memory banks, and use a loss that explicitly pushes the embeddings of dissimilar images away from each other. They often require a mining procedure to search for offending dissimilar samples from a memory bank or from the current batch. Quantization-based approaches force the embeddings of different samples to belong to different clusters on the unit sphere. Collapse is prevented by ensuring that the assignment of samples to clusters is as uniform as possible. A similarity term encourages the cluster assignment score vectors from the two branches to be similar. More recently, a few methods have appeared that do not rely on contrastive samples or vector quantization, yet

produce high-quality representations. **They exploit several tricks:** batch-wise or feature-wise normalization, a "momentum encoder" in which the parameter vector of one branch is a low-pass-filtered version of the parameter vector of the other branch or a stop-gradient operation in one of the branches. The dynamics of learning in these methods, and how they avoid collapse, is not fully understood, although theoretical and empirical studies point to the crucial importance **of batch-wise or feature-wise normalization**. The last alternative class of collapse prevention methods relies on maximizing the information content of the embedding. These methods prevent informational collapse by decorrelating every pair of variables of the embedding vectors. This indirectly maximizes the information content of the embedding vectors.

Another example of joint architecture is the **Siamese Networks** [18]. Siamese Models have a typical 'two towers' architecture. Each tower processes a view of a data sample. Considering the natural similarity between the two views of the same sample, the encoded representations in the high-dimensional latent space should be close to each other. Hence, during training, the representations from one tower can be seen as the training target, i. e., pseudo-labels, for the other tower. The neural encoders of both towers share the same or similar architecture – their parameters can be shared or independent. When taking negative samples in the training objectives, a contrastive loss is formulated which pulls the representations of different views from the same data close together, while pushing the one from negative samples far away. The model can be optimized by applying standard backpropagation. However, without using negative samples, the Siamese model is prone to mode collapse, i. e., when the model's output is very similar (or even identical) for different inputs.

Another approach recently trending is **Auto-encoding** [19]. It is based on the use of auto-encoders. A standard auto-encoder learns a compressed latent embedding that represents the input of the encoder and expects to reconstruct the original input from the latent representation, i.e., the decoder output. The dimensionality of the latent representation must be carefully designed, as it determines the representation reliability. When setting a too large latent dimensionality, an auto-encoder risks learn an identity function, i. e., maps the input directly to the output, and hence becomes useless. Various techniques to prevent auto-encoders from learning an identity function do exist, e.g., denoising auto-encoders.

3.3 Why to use Self-Supervised Learning for AST

Self-supervised learning is a learning mode that involves training a model on a dataset without explicitly providing labels or supervision for each example in the dataset. Instead, the model is provided with a set of "tasks" that it needs to learn to solve, and it is given a set of labeled examples that it can use to learn the tasks.

Self-supervised learning has several advantages for audio classification, particularly when the amount of labeled data available for training is limited. Here are a few reasons why self-supervised learning might be preferred for audio classification:

- **It can be more efficient**: Self-supervised learning can be more efficient than supervised learning, as it does not require manually labeling each example in the dataset. This can be particularly useful when the amount of labeled data available is limited, as it allows the model to make use of more of the data for training.
- **It can learn from a wider range of data**: Self-supervised learning allows the model to learn from a wider range of data, as it does not require explicit labels for each example. This can be useful for audio classification, as it allows the model to learn from more diverse audio data and potentially improve its performance.
- **It can learn useful features for downstream tasks:** Self-supervised learning allows the model to learn useful features and representations of the data that can be useful for other tasks. This can be particularly useful for audio classification, as the learned features may be useful for other audio processing tasks, such as speech recognition or audio segmentation.

Anyway, Self-Supervised learning is not always the best choice for audio classification, and the suitability of self-supervised learning will depend on the specific characteristics of the data and the task at hand. However, self-supervised learning can be a useful approach to consider when training a model for audio classification, particularly when the amount of labeled data is limited.

3.4 Related Works

3.4.1 SSAST

SSAST[20] (Self-Supervised AST) proposes a self-supervised framework for the original AST model called MSPM: a novel patch-based joint discriminative and generative self-supervised learning framework. MSPM is a patch-based self-supervised learning framework in audio and speech domains.
Thanks to the design of AST, they can mask spectrogram patches rather than the entire time frame during pretraining, which allows the model to learn both the temporal and frequency structure of the data. In addition, it is used a cluster factor to control how masked patches cluster. The model is forced to learn more about global spectrogram structure with a larger cluster factor, and more local structure with a smaller one. Specifically, this is considered the first work to use both discriminative and generative masked modeling for pretraining.

3.4.2 HTS-AT

HTS-AT [19] is a hierarchical audio transformer with a token-semantic module for audio classification based on AST. It is composed of two key designs: a hierarchical transformer structure and a window attention mechanism. In this work to better capture the relationship among frequency bins of the same time frame, we first split the Mel-spectrogram into patch windows and then split the patches inside each window. Then the patch tokens are sent into several groups of transformer-encoder

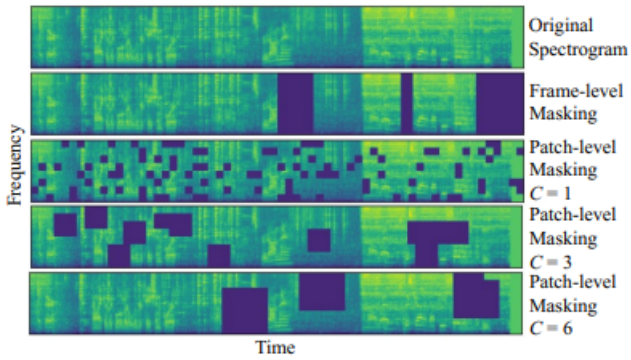


Figure 3.2. By utilizing patch-level masking with varying cluster factors C, the model can gain a deeper understanding of both global and local spectrogram structure. As C increases, the model focuses on learning more about the overall structure of the spectrogram, while a smaller C shifts the focus to local structures. To ensure that the model is exposed to both global and local structures, we implement random C during the pretraining phase. This approach, in contrast to frame-level masking SSL methods that focus solely on temporal frame structure, allows the model to also learn about the frequency structure of the spectrogram.

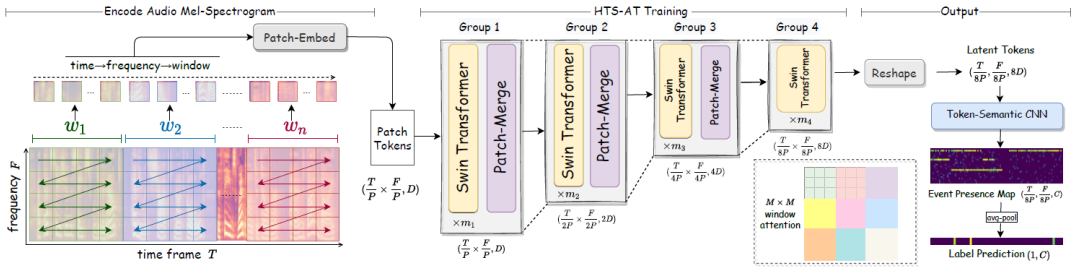


Figure 3.3. HTS-AT Architecture

blocks. At the end of each group, it is added a Patch-Merge layer to reduce the sequence size. For each transformer block inside the group, they adopted a window attention mechanism to reduce the calculation. Finally, as the network goes deeper, the Patch-Merge layer will merge adjacent windows, thus the attention relation is calculated in a larger space.

3.4.3 MAE-AST

Masked Autoencoding Audio Spectrogram Transformer (MAE-AST)[19], runs only unmasked tokens through a large encoder and then concatenates mask tokens with the encoder output embeddings before feeding them to a shallow decoder. It is another interesting solution for self-supervised learning based on AST. Key features of this work's architecture are the use of two different loss functions and of the joint use of Discriminative and Generative pretraining like SSAST.

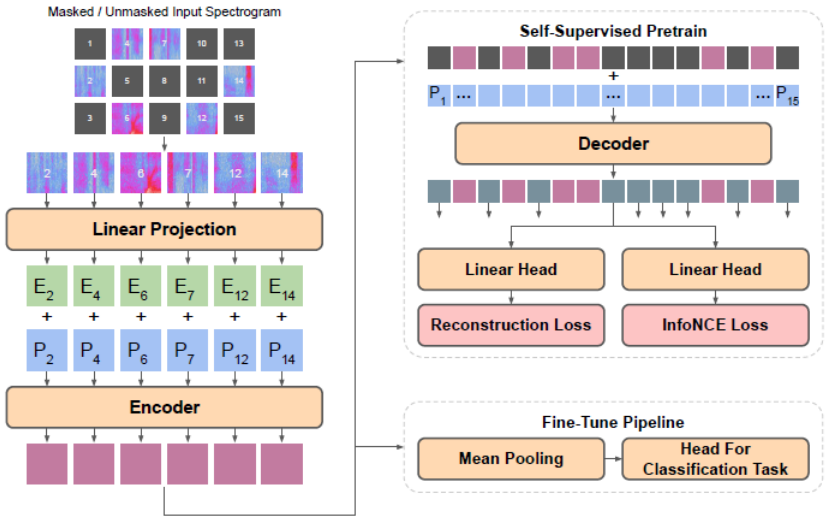


Figure 3.4. The MAE-AST model utilizes a patch-based input architecture where the input spectrogram, with dimensions of 128x100t, is divided into 16x16 pixel patches. Before training, a mask is applied to select tokens. The unmasked patches are unrolled by channel dimension, then by time dimension, and transformed into a vector via linear projection. These patch embeddings are combined with 1D sinusoidal positional embeddings and fed into the encoder. During pretraining, the masked tokens are paired with the encoder outputs and positional embeddings are added to all tokens before being processed by the decoder. The model calculates a joint discriminative and generative loss on the decoder outputs for masked tokens. For fine-tuning, the encoder outputs are mean pooled to generate a representation vector, and the decoder remains untouched.

3.4.4 AsiT

ASiT [21] is a self-supervised framework based on vision transformers for general audio representations. This work is based on GMML **(Group Masked Model Learning)**. The main idea of GMML is to corrupt a group of connected patches representing a "significant" part of a given visual input and recover them by learning a model. The the underlying hypothesis is that, recovering the corrupted parts from the uncorrupted parts based on the context from the whole visual field, the network will implicitly learn the the notion of visual integrity. Intuitively the network will only be able to recover the missing information if it learns the characteristic properties of visual stimuli corresponding to specific actions impacting the visual input. For spectrogram reconstruction, we propose to use the vision transformer as a group-masked autoencoder, i.e. visual transformer autoencoder with group masked model learning. By analogy to autoencoders, our network is trained to reconstruct the input spectrogram through the output tokens of the transformer.

3.5 Transformer on AST

The Transformer model architecture presented in this paper utilizes an attention mechanism to establish global connections between input and output. This approach allows for increased parallelization and significantly enhances translation

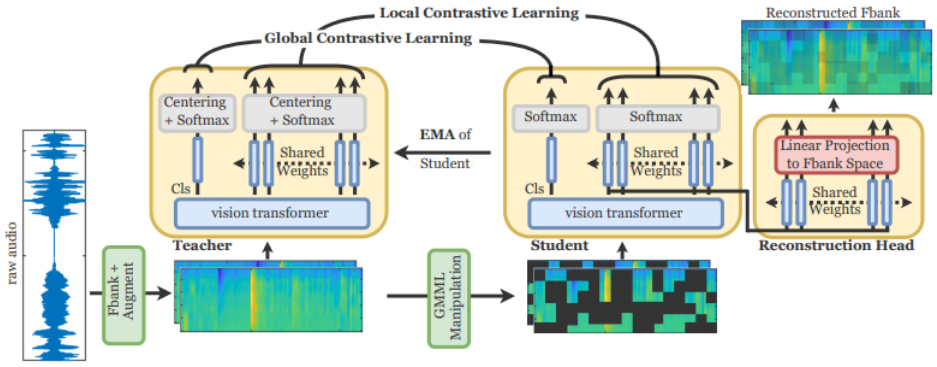


Figure 3.5. The ASiT self-supervised framework is designed to enhance the understanding of audio spectrograms. The process begins by taking a 10-second audio spectrogram and creating two random augmented views, each lasting 6 seconds. These clean spectrograms are then passed through a GMML-based manipulation block to produce corrupted versions. These clean and corrupted spectrograms are then fed into the teacher and student networks, respectively. The recovery of the transformed information from the non-transformed class-token and data-tokens suggests that the network has successfully learned the semantics of both the local and global representations of the audio, as well as gained useful inductive bias by studying the local statistical correlations within the spectrogram.

quality. Typically, neural sequence transduction models have an encoder-decoder structure, where the encoder converts an input sequence of symbols into a continuous representation and the decoder generates an output sequence. The Transformer follows this structure, utilizing stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. The authors chose to use self-attention based on three criteria: total computational complexity per layer, the ability to parallelize computation, and efficient path length between dependencies in the network. The Transformer can utilize two types of attention functions: Scaled Dot-Product Attention, which calculates attention simultaneously for multiple queries in a matrix, and Multi-head attention, which allows the model to consider information from different positions and subspaces. Compared to recurrent layers, self-attention layers have a constant number of sequential operations and are faster in terms of computational complexity when the sequence length is smaller than the representation dimensionality, which is commonly the case in machine translations.

Other applications

The transformer has been born for transduction model tasks, anyway, a lot of other applications can be applied, like Speech Recognition, Computer Vision, Generative Models, Time Series prediction, and more.

Chapter 4

Proposed Joint Embedding AST Network

4.1 VCR-AST

Recent self-supervised methods for image representation learning aim at increasing the agreement between embedding vectors produced by encoders fed with different views of the same image. Other approaches rely on a joint embedding architecture, such as the Siamese network architecture, where two networks are trained to produce similar embeddings for different views of the same image. The main challenge with joint embedding architectures is preventing collapse, where the two branches ignore the inputs and produce identical and constant output vectors. Two main approaches to preventing collapse are contrastive methods and information maximization methods.
However, a main challenge in these methods is preventing the collapse. To address this issue, we propose VCR-AST (Variance-Invariance-Covariance Regularization Self-Supervised for AST), a method that explicitly avoids the collapse problem by applying two regularization terms to both embeddings separately:

- A term that maintains the variance of each embedding dimension above a threshold
- A term that decorrelates each pair of variables

Unlike othe approaches, VCR-AST does not require techniques such as weight sharing, batch normalization, feature-wise normalization, output quantization, stop gradient, or memory banks. It also achieves results on par with state-of-the-art methods on several downstream tasks. Additionally, the variance regularization term stabilizes the training of other methods and leads to performance improvements.
Some The main advantages of VCR-AST include:

- **Improved generalization:** by training a model to predict the transformation of an input data point, VCR-AST can improve the model's ability to generalize to unseen data.
- **Increased data efficiency:** VCR-AST can make more effective use of limited data by using self-supervised learning, which allows the model to learn from the input data itself in a faster way.
- **Enhanced representation learning:** VCR-AST can help a model learn more robust and discriminative representations of the input data, which can improve performance on downstream tasks.
- **Increased flexibility:** VCR-AST can be applied to a wide range of data types and can be easily integrated into existing machine learning pipelines.

4.2 Intuition

AST reaches great results but a huge amount of time and data to pretrain are required. In order to train the net faster and with only audio datasets, we decided to apply a new self-supervised learning framework to the net. The core concept of this

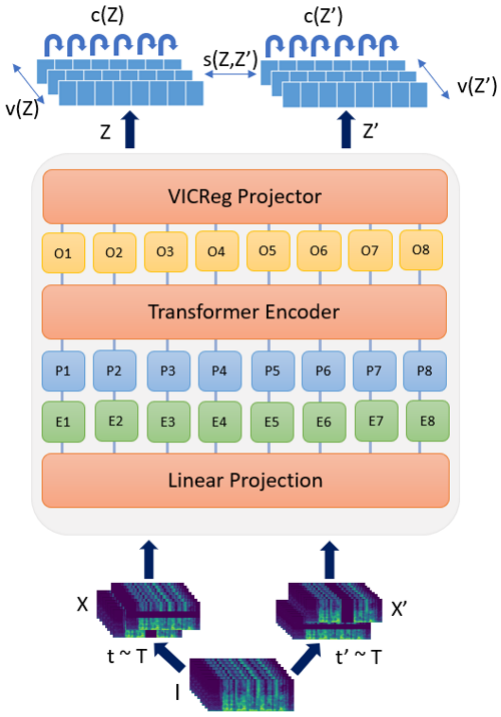


Figure 4.1. VCR-AST: joint embedding architecture with variance, invariance, and covariance regularization. Given a batch of images I, two batches of different transformed views X and X_0 are produced and are then encoded into representations Y and Y'. The representations are fed to an expander producing the embeddings Z and Z' . The distance between two embeddings from the same image is minimized, and the variance of each embedding variable over a batch should not decrease so much during the training, instead covariance between pairs of embedding variables in a batch tends to zero, decorrelating the variables from each other. Although the two branches do not require identical architectures nor share weights, in most of our experiments, they are Siamese with shared weights: the encoders are AST model backbones with output dimension 1024. The expanders have 3 fully-connected layers of size 4096.

approach is to utilize a loss function that comprises three components: Invariance, Variance, and Covariance.

- **Invariance** term calculates the mean square distance between the embedding vectors to ensure that similar samples are close in the embedding space.
- **Variance** term uses a hinge loss to maintain a standard deviation above a certain threshold for each variable of the embedding, thus ensuring that samples within a batch are different.
- **Covariance** term attracts the covariances between every pair of centered embedding variables towards zero, preventing correlated variables and an informational collapse.

Both Variance and Covariance terms are applied to both branches of the architecture separately, preserving the information content of each embedding. This

method is more generally applicable than previous methods as it does not require shared weights, identical architectures, or similar inputs. Furthermore, it allows for non-contrastive self-supervised joint embedding for multi-modal signals and has been shown to improve training stability and performance on downstream tasks. Overall, this is an explicit, effective, and simple method for preventing a collapse in self-supervised joint-embedding learning.
Additionally, this method does not require the use of a memory bank, contrastive samples, or large batch size, making it more practical and efficient. It also does not require batch-wise or feature-wise normalization, making it more robust and less sensitive to data variations. Overall, this method is a significant step forward in the field of self-supervised joint-embedding learning, as it provides a simple yet effective solution to the problem of informational collapse.

4.3 Methodology

Given an image *i* sampled from a dataset **D**, two transformations **t** and **t'** are sampled from a distribution T to produce two different views *x = t(i)* and *x' = t'(i)* of *i*. We apply these three* transformations to each sample with two different random setups in order to train the model with two different views of the same sample:

- **FrequencyMasking** Apply masking to a spectrogram in the frequency domain.
- **TimeMasking** Apply masking to a spectrogram in the time domain.
- **TimeStretching** Stretch STFT in time without modifying the pitch for a given rate. (*future work)

The parameter is randomized each time but kept inside a certain range: FrequencyMasking (1,200), TimeMasking (1,200), and TimeStretching (0.8,1.2). The views *x* and *x'* are first encoded by f_θ into their representations $y = f_\theta(x)$ and $y' = f_\theta(x)$, which are then mapped by the expander h_ϕ onto the embeddings $z = h_\phi(y)$ and $z' = h_\phi(y')$. The loss is computed at the embedding level on *z* and *z'*. We describe here the variance, invariance, and covariance terms that compose our loss function. The audio samples are processed in batches, and we denote $Z = [z_1, ..., z_n]$ and $Z' = [z'_1, ..., z'_n]$ the two batches composed of n vectors of dimension d, of embeddings coming out of the two branches of the siamese architecture. We denote by z^j the vector composed of each value at dimension *j* in all vectors in *Z*. We define the variance regularization term v as a hinge function on the standard deviation of the embeddings along the batch dimension:

$$v(Z) = \frac{1}{d}\sum_{j=1}^{d} max(0, \gamma - S(z^j, \epsilon)), \tag{4.1}$$

where S is the regularized standard deviation defined by:

$$S(x, \epsilon) = \sqrt{Var(x) + \epsilon}, \tag{4.2}$$

γ is a constant target value for the standard deviation, fixed to 1 in our experiments, and ϵ is a small scalar preventing numerical instabilities. This criterion encourages the variance inside the current batch to be equal to γ along each dimension, preventing collapse with all the inputs mapped on the same vector. Using the standard deviation and not directly the variance is crucial. Indeed, if we take $S(x) = Var(x)$ in the hinge function, the gradient of S with respect to x becomes close to 0 when x is close to $\bar{a}$. In this case, the gradient of v also becomes close to 0 and the embeddings collapse. We define the covariance matrix of Z as:

$$C(Z) = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - \bar{z})(z_i - \bar{z})^T, \text{where } \bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i \tag{4.3}$$

we can then define the covariance regularization term c as the sum of the squared off-diagonal coefficients of $C(Z)$, with a factor $1/d$ that scales the criterion as a function of the dimension:

$$c(Z) = \frac{1}{d}\sum_{i \neq j}[C(Z)]_{i,j}^2 \tag{4.4}$$

This term encourages the off-diagonal coefficients of C(Z) to be close to 0, decorrelating the different dimensions of the embeddings and preventing them from encoding similar information. Decorrelation at the embedding level ultimately has a decorrelation effect at the representation level, which is a non-trivial phenomenon. We finally define the invariance criterion s between Z and Z' as the mean-squared euclidean distance between each pair of vectors, without any normalization:

$$s(Z, Z') = \frac{1}{\sum_i ||z_i - z_i'||_2^2} \tag{4.5}$$

The overall loss function is a weighted average of the invariance, variance, and covariance terms:

$$l(Z, Z') = \lambda s(Z, Z') + \mu[v(Z) + v(Z')] + \nu[c(Z) + c(Z')] \tag{4.6}$$

where λ,μ, and ν are hyper-parameters controlling the importance of each term in the loss. In our experiments, we set $\nu = 1$ and perform a grid search on the values of λ and μ with the base condition $\lambda = \mu > 1$. The overall objective function taken on all images over an unlabelled dataset D is given by:

$$L = \sum_{I \in D}\sum_{t,t' \sim T} l(Z^I, Z'^I) \tag{4.7}$$

where Z^I, Z'^I are the batches of embeddings corresponding to the batch of images I transformed by t and t'. The objective is minimized for several epochs, over the encoder parameters θ and expander parameters ϕ

4.3.1 Implementation details

Backbone Architecture

The backbone is AST a convolution-free, a purely attention-based model with this configuration:

Layer	Description
PatchEmbed	Embeds image patches into a high-dimensional space using convolution with a kernel size of $(16, 16)$ and a stride of $(16, 16)$.
Dropout	A regularization layer that randomly sets input elements to zero with a probability of 0 during training.
ModuleList	A list of 4 submodules that can be indexed like a regular Python list.
Block	A block of the transformer architecture, consists of attention, normalization, and multi-layer perceptron (MLP) layers.
LayerNorm	A layer that normalizes the mean and variance of its inputs with a default epsilon of $1e-06$ and element-wise affine transformation.
Attention	An attention mechanism for the transformer architecture with a linear layer of size $(768, 2304)$ for computing query, key, and value vectors, and a linear layer of size $(768, 768)$ for projecting the attention output.
Linear	A fully-connected linear layer.
Identity	An identity layer that returns its input unchanged.
Mlp	A multi-layer perceptron with a linear layer of size $(768, 3072)$, a nonlinear activation function (GELU), a linear layer of size $(3072, 768)$, and dropout with a probability of 0.
VCR	layer for variance-covariance regularization contains a submodule projector consisting of
BatchNorm1d	A batch normalization layer with input size 4096. The default epsilon is $1e-05$, the momentum is 0.1, and the affine transformation and tracking of running statistics are enabled.
ReLU	A nonlinear activation function that sets negative input elements to zero.
Linear	A fully-connected linear layer with input size 4096 and output size 4096. This layer does not have a bias.

Table 4.1. VCR-AST configuration

Loss Computation paramters

For what concerns the computation of the loss functions we use three parameters:

- λ (similarity coefficient) = 25.0
- μ (standard deviation/variance coefficient) = 25.0
- ν (covariance coefficient) = 1.0

LARS Optimizer

LARS (Layer-wise Adaptive Rate Scaling) is an optimization algorithm that is designed to optimize deep neural networks efficiently by adapting the learning rate of each parameter independently. It is particularly useful for training very deep networks, where the gradients can become small or vanish altogether, leading to slow or stalled learning.

In LARS, the learning rate for each parameter is scaled by the ratio of the parameter's L2 norm to the L2 norm of the gradient. This helps to prevent the gradients from becoming too small, allowing the network to continue learning at a reasonable pace. Additionally, LARS can adapt the learning rate of each parameter based on the magnitude of its gradients, allowing it to handle the vanishing and exploding gradient problem more effectively.

LARS has been shown to achieve good results on a variety of tasks, including image classification and language modeling. It is often used in conjunction with the Adam optimizer, which can provide faster convergence and better generalization.

Loss function Code

Algorithm 1: VCR

```
class VICReg(nn.Module):
def ___init___(self, embeddings=1024):
super().___init___()
self.num_features = int(mlp.split("-")[-1])
self.projector = Projector(768) # a linear 3-layers projector
def forward(self, x, y):
B = x.shape[0]
tot_loss = 0
x = self.projector(x) # pass x and y through the projector
y = self.projector(y)
repr_loss = F.mse_loss(x, y) #compute representation loss
x = x - x.mean(dim=0)
y = y - y.mean(dim=0)
std_x = torch.sqrt(x.var(dim=0) + 0.0001)
std_y = torch.sqrt(y.var(dim=0) + 0.0001)
std_loss = torch.mean(F.relu(1 - std_x)) / 2 + torch.mean(F.relu(1 -
  std_y)) / 2 #compute std loss
cov_x = (x.T @ x) / (B - 1)
cov_y = (y.T @ y) / (B - 1)
cov_loss = off_diagonal(cov_x).pow_(2).sum().div( self.num_features ) +
  off_diagonal(cov_y).pow_(2).sum().div(self.num_features) #compute
  covariance loss
return (sim_coeff * repr_loss + std_coeff * std_loss + cov_coeff *
  cov_loss),[round(repr_loss.item(),3), round(std_loss.item(),3),
  round(cov_loss.item(),3)]
where :
mlp = "4096-4096-4096"
sim_coeff = 25.0
std_coeff = 25.0
cov_coeff = 1.0
```

Algorithm 2: Projector

```
def Projector(embedding):
mlp_spec = f"embedding-mlp"
layers = []
f = list(map(int, mlp_spec.split("-")))
for i in range(len(f) - 2):
layers.append(nn.Linear(f[i], f[i + 1]))
layers.append(nn.BatchNorm1d(f[i + 1]))
layers.append(nn.ReLU(True))
layers.append(nn.Linear(f[-2], f[-1], bias=False))
return nn.Sequential(*layers)
```

Chapter 5

Experimental Results

5.1 Datasets

Recent years have brought a steady stream of advances in machine perception. Computer systems undertake progressively more complex tasks, at times even surpassing human capabilities. A significant part of these spectacular achievements has come in visual recognition, with the recent proliferation of successful deep learning approaches. At the same time, research in auditory recognition tasks has been focusing mostly on speech and music processing. Analysis of environmental sounds (a very diverse group of everyday audio events that cannot be described as speech or music) has lagged behind in applying those recent improvements, despite numerous possible applications in audio surveillance systems, hearing aids, smart room monitoring, and video content highlight generation. One of the objective impediments to more active research in this field is strong fragmentation and difficulty in comparability and reproducibility. This scarcity of publicly available datasets and difficulty in accessing the original code for study replication make research reproducibility efforts harder than they should be. Therefore a lot of works are based on image datasets and pre-trained models.

5.1.1 ESC50

The dataset consists of sound clips constructed from recordings available publicly through the Freesound project. Classes included in the labeled part of the dataset were arbitrarily selected to maintain a balance between major types of sound events, all the while taking into consideration the limitations in the number and diversity of available source recordings, and subjectively assessing the usefulness and distinctiveness of each class. The Freesound database of field recordings was queried for common terms related to the constructed classes. Search results were individually evaluated and verified by the author by annotating fragments containing events belonging to the given class. These annotations were then used to extract 5-second-long recordings of audio events (shorter events were padded with silence as needed). The extracted samples were reconverted to a unified format (44.1 kHz, single channel, Ogg Vorbis compression at 192 kbit/s). The labeled datasets were consequently arranged into 5 uniformly-sized cross-validation folds, ensuring that clips originating from the same initial source files are always contained in a single fold.
The ESC-50 dataset consists of 2 000 labeled environmental recordings equally balanced between 50 classes (40 clips per class). For convenience, they are grouped into 5 loosely-defined major categories (10 classes per category):

- animal sounds
- natural soundscapes and water sounds
- human (non-speech) sounds
- interior/domestic sounds
- exterior/urban noises

Animals	Natural soundscapes & water sounds	Human, non-speech sounds	Interior/domestic sounds	Exterior/urban noises
Dog	Rain	Crying baby	Door knock	Helicopter
Rooster	Sea waves	Sneezing	Mouse click	Chainsaw
Pig	Crackling fire	Clapping	Keyboard typing	Siren
Cow	Crickets	Breathing	Door, wood creaks	Car horn
Frog	Chirping birds	Coughing	Can opening	Engine
Cat	Water drops	Footsteps	Washing machine	Train
Hen	Wind	Laughing	Vacuum cleaner	Church bells
Insects (flying)	Pouring water	Brushing teeth	Clock alarm	Airplane
Sheep	Toilet flush	Snoring	Clock tick	Fireworks
Crow	Thunderstorm	Drinking, sipping	Glass breaking	Hand saw

The dataset provides exposure to a variety of sound sources - some very common (laughter, cat meowing, dog barking), some quite distinct (glass breaking, brushing teeth) and then where the differences are more nuanced (helicopter and airplane noise). One of the possible deficiencies of this dataset is the limited number of clips available per class.

5.1.2 Speechcommands

The Speech Commands dataset is an attempt to build a standard training and evaluation dataset for a class of simple speech recognition tasks. Its primary goal is to provide a way to build and test small models that detect when a single word is spoken, from a set of ten or fewer target words, with as few false positives as possible from background noise or unrelated speech. This task is often known as keyword spotting. Many voice interfaces rely on keyword spotting to start interactions. For example, you might say "Hey Google" or "Hey Siri" to begin a query or command for your phone. Once the device knows that you want to interact, it's possible to send the audio to a web service to run a model that is only limited by commercial considerations, since it can run on a server whose resources are controlled by the cloud provider. The initial detection of the start of an interaction is impractical to run as a cloud-based service though since it would require sending audio data over the web from all devices all the time. This would be very costly to maintain and would increase the privacy risks of the technology.
Instead, most voice interfaces run a recognition module locally on the phone or other device. This listens continuously to audio input from microphones, and rather than sending the data over the internet to a server, they run models that listen for the desired trigger phrases.

This Speech Commands dataset aims to meet the special needs around building and testing on-device models, to enable model authors to demonstrate the accuracy of their architectures using metrics that are comparable to other models, and give a simple way for teams to reproduce baseline models by training on identical data. The hope is that this will speed up progress and collaboration, and improve the overall quality of the available models. A second important audience is hardware manufacturers. By using a publicly-available task that closely reflects product requirements, chip vendors can demonstrate the accuracy and energy usage of their

offerings in a way that's easily comparable for potential purchasers. This increased transparency should result in hardware that better meets product requirements over time. The models should also provide clear specifications that hardware engineers can use to optimize their chips, and potentially suggest model changes that make it easier to provide efficient implementations. This kind of co-design between machine learning and hardware can be a virtuous circle, increasing the flow of useful information between the domains in a way that helps both sides.

The dataset is composed of about 100k samples of 1 second of English keywords divided into 35 classes as follows:

Word	Number of Utterances
Backward	1,664
Bed	2,014
Bird	2,064
Cat	2,031
Dog	2,128
Down	3,917
Eight	3,787
Five	4,052
Follow	1,579
Forward	1,557
Four	3,728
Go	3,880
Happy	2,054
House	2,113
Learn	1,575
Left	3,801
Marvin	2,100
Nine	3,934
No	3,941
Off	3,745
On	3,845
One	3,890
Right	3,778
Seven	3,998
Sheila	2,022
Six	3,860
Stop	3,872
Three	3,727
Tree	1,759
Two	3,880
Up	3,723
Visual	1,592
Wow	2,123
Yes	4,044
Zero	4,052

5.1.3 LibriSpeech

The LibriSpeech corpus is derived from audiobooks that are part of the LibriVox project and contains 1000 hours of speech sampled at 16 kHz. The dataset can be used to train a model for Automatic Speech Recognition (ASR). The model is presented with an audio file and asked to transcribe the audio file to written text. The most common evaluation metric is the word error rate (WER).

The audio is in English. There are two configurations: clean and other. The speakers in the corpus were ranked according to the WER of the transcripts of a model trained on a different dataset, and were divided roughly in the middle, with the lower-WER speakers designated as "clean" and the higher WER speakers des-

ignated as "other".

A typical data point comprises the path to the audio file, usually called the file, and its transcription, called text. Some additional information about the speaker and the passage which contains the transcription is provided.

```
{'chapter_id': 141231,
 'file': '/home/patrick/.cache/huggingface/datasets/downloads/
  'audio': {'path': '/home/patrick/.cache/huggingface/datasets
  'array': array([-0.00048828, -0.00018311, -0.00137329, ...,
          0.00091553,  0.00085449], dtype=float32),
  'sampling_rate': 16000},
 'id': '1272-141231-0000',
 'speaker_id': 1272,
 'text': 'A MAN SAID TO THE UNIVERSE SIR I EXIST'}
```

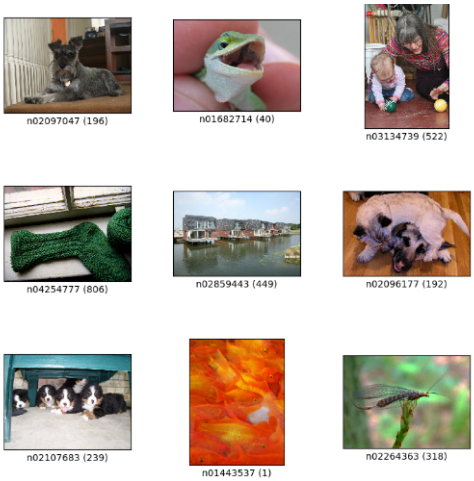
The size of the corpus makes it impractical, or at least inconvenient for some users, to distribute it as a single large archive. Thus the training portion of the corpus is split into three subsets, with approximate sizes of 100, 360, and 500 hours respectively. A simple automatic procedure was used to select the audio in the first two sets to be, on average, of higher recording quality and with accents closer to US English. An acoustic model was trained on WSJ's si-84 data subset and was used to recognize the audio in the corpus, using a bigram LM estimated on the text of the respective books. We computed the Word Error Rate (WER) of this automatic transcript relative to our reference transcripts obtained from the book texts. The speakers in the corpus were ranked according to the WER of the WSJ model's transcripts and were divided roughly in the middle, with the lower-WER speakers designated as "clean" and the higher-WER speakers designated as "other".

5.1.4 ImageNet

This dataset is basically used to pretrain AST when using supervised learning. This task results are necessary since most Audio Datasets are not large enough to train effectively the transformer and the solution adopted is to pretrain the model on this dataset and then use transfer learning to train the model on smaller audio datasets (esc50, speechcommands, etc.). The ImageNet dataset contains 14,197,122 annotated images according to the WordNet hierarchy divided into 1000 classes. Since 2010 the dataset is used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a benchmark in image classification and object detection. The publicly released dataset contains a set of manually annotated training images. A set of test images is also released, with the manual annotations withheld. ILSVRC annotations fall into one of two categories:

- Image-level annotation of a binary label for the presence or absence of an object class in the image, e.g., “there are cars in this image” but “there are no tigers,”.
- Object-level annotation of a tight bounding box and class label around an

object instance in the image, e.g., “there is a screwdriver centered at position (20,25) with a width of 50 pixels and height of 30 pixels”.



5.2 Pretraining Configuration

For pretraining AST with our new proposed method, during the experiments, we have used Batch Sizes of 6, 8, and 12, an initial learning rate of 1e-4, and cutting the learning rate with a ***ReduceLROnPlateau*** scheduler into half if the pretext task performance on the validation set stops improving for 2 epoch (in our case the metric to optimize is the loss so in the scheduler.step() function we will pass ***-loss***). We Optimize the network using the LARS optimizer previously cited with learning_rate 0 and weight decay 1e-6. Also, we introduce the use of a Scaler: ***GradScaler***. This is used to prevent underflow, so “gradient scaling” multiplies the network’s loss(es) by a scale factor and invokes a backward pass on the scaled loss(es). Gradients flowing backward through the network are then scaled by the same factor. In other words, gradient values have a larger magnitude, so they do not flush to zero. We pretrain VCR-AST on Google Colab which provides (for a small amount of time) an NVIDIA T4 with 16 GB. The pretraining takes about 2 days for ESC50 and for LibriSpeech+ESC50 would take approximately 15 days. In addition, every dataset used has a different configuration for Mean, standard deviation, target length, and noise as described in table 5.1

Finally, the proposed method configuration can be divided into two main branches:

- **Frame-based:** audio signal is divided into overlapping or non-overlapping frames, and the spectrogram is computed for each frame independently. This can be useful for analyzing short-term changes in the frequency content of the signal, but may not capture longer-term patterns or relationships in the data.
- **Patch-based:** audio signal is divided into overlapping patches and the spectrogram is computed for each patch. This can be more effective at capturing longer-term patterns in the data, but may not capture as much detail about short-term changes in the signal.

Dataset	Mean	Std	Target Length	Noise
Librispeech	-4.2677393	4.5689974	1024	False
ESC50	-6.6268077	5.358466	512	False
SpeechCommands	-6.845978	5.5654526	128	True

Table 5.1. Datasets configuration

5.2.1 Pretraining Performance

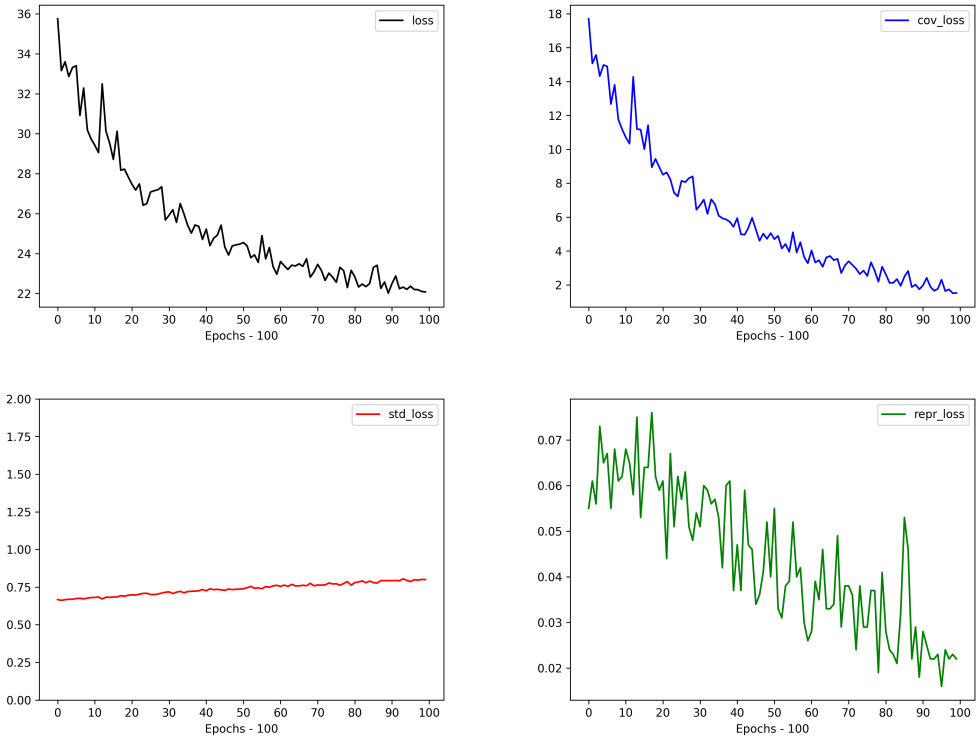


Figure 5.1. Train loss of the proposed method

We pretrain the model with a base model size, a number of 250 masked patches, and a batch size of 12. In the figures is shown the training loss. As described in the previous chapter it is the weighted sum of three subcomponents: Covariance Loss, obtained from the covariances matrices of x,y (the two different representations of the sample), the Std Loss, obtained from the mean of the standard deviation of x,y, the Representation Loss, obtained as the mean squared error of x,y projected. In these figures, we present the trend during the epochs of the 3 components and the total error. The method aims at minimizing the distance between the two embeddings that come from the same image (repr_loss) and to maintain a certain level of variance for each embedding variable in a batch (std_loss). Additionally, the algorithm tries to reduce the covariance (cov_loss) between the pair of represen-

tations over the batch, which would decorrelate the samples from each other. The method can be implemented using two branches, which do not have to be identical or share weights. However, in most of the experiments described, the two branches are similar to Siamese network architecture.

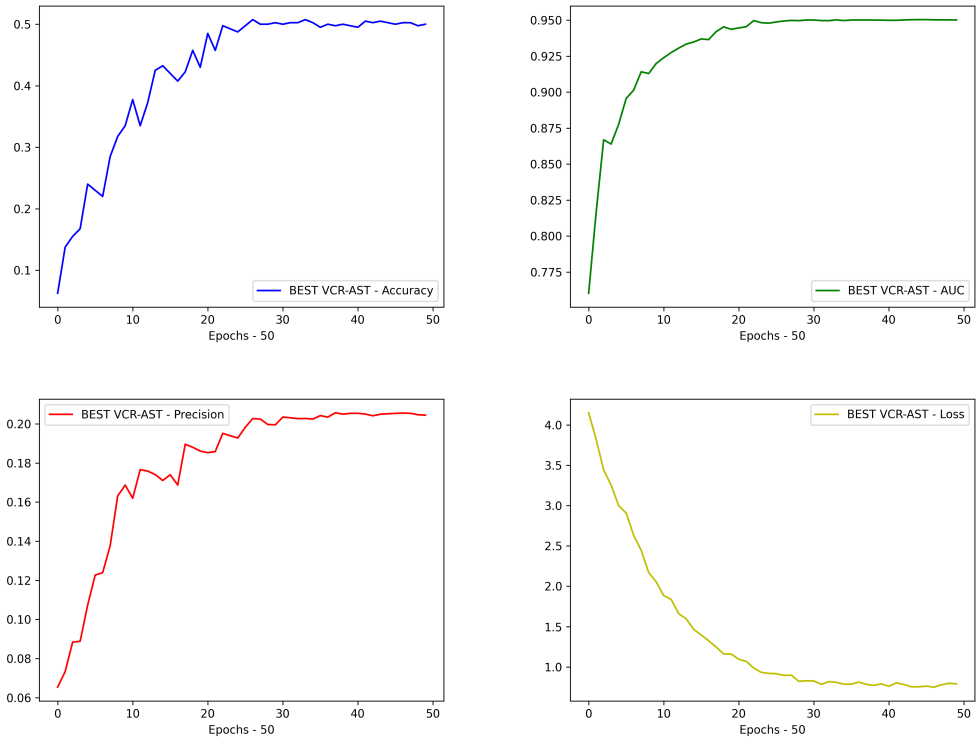


Figure 5.2. Best pretrained VCR-AST model: pretrained for 200 epochs (26k iterations) finetuned for 50 epochs (7k iterations), here we present the accuracy trend (1), AUC trend (2) that provides an aggregate measure of performance across all possible classification thresholds, AP (3), Train loss trend (4)

5.2.2 Pretraining Timing Comparison

Another metrics that we have compared is the Training time: we have pretrained the model, again, with the same parameters, and the results are shown in the Table 5.2.
The results show that VCR-AST is almost 3 times faster than SSAST for every time metrics. This also results in a *"Time for epoch"* of 4 minutes for SSAST against less than 2 minutes for VCR-AST.

5.3 Evaluation Configuration

We evaluate the pretrained model on 3 commonly used audio and speech benchmarks. We use SpeechCommands and ESC50 as benchmarks (the last one especially for resources reasons, only ESC50 results are reported) and keep the same setting

Model	Per Sample Total Time	P.S. Data Time	P.S. DNN Time
SSAST	0.14529	0.03311	0.11218
VCR-AST	**0.06278**	**0.01968**	**0.04310**

Table 5.2. Time comparison between SSAST and VCR-AST

intentionally to make a fair comparison with AST and SSAST. To further evaluate the model performance on downstream speech tasks and compare with previous self-supervised models that focus on speech, we test the pretrained AST also on Librispeech.
ESC-50 We use the ESC-50 dataset for the single-label audio event classification task. ESC-50 is an audio classification dataset consisting of 2,000 5-second environmental audio recordings organized into 50 classes.
Speech Commands V2 (KS2) We use the Speech Commands V2 (Warden 2018) for the keyword spotting task. The Speech Command V2 dataset consists of 105,829 1-second recordings of 35 common speech commands.
LibriSpeech (LS) We use this dataset that contains hours of speech random speakers for the speaker identification task. The task goal is to classify each utterance by its speaker identity where speakers are in the same predefined set for both training and testing.

5.4 Finetuning Configuration

To make a fair comparison with previous work, for the Librispeech, ESC-50, and Speechcommands V2 experiments, we train and evaluate the model using the exact same training and evaluation settings as the original AST. For ESC50 we use SpecAugment, initial learning rate 1e-4, 50 epochs. While in supervised models of AST, we use the output of CLS token as the audio clip representation, in this self-supervised case we use mean pooling as supervision is given to individual tokens in pretraining, keeping pretraining and fine-tuning consistent can improve the performance and make the comparison fairer.

5.5 Performance on Downstream tasks

To compare performances between our proposed methodology and state-of-the-art method SSAST, we pretrain the model in the four configurations shown (SSAST-Patch, SSAST-Frame, VCR-AST Patch, VCR-AST Frame) for 100 epochs, with the same parameters of initial learning rate (1e-4), batch-size (12), model-size (base). The model is pretrain and tested on ESC-50. Finally, they are fine-tuned for 50 epochs. The behavior and results are almost the same with a small increase in accuracy in favor of our proposed method **(Figure 5.3)**.

Model	Accuracy	AP	AUC
SSAST-Patch	0.31	0.11	0.90
SSAST-Frame	0.32	**0.14**	**0.91**
VCR-AST-Patch	0.33	0.13	0.90
VCR-AST-Frame	**0.35**	0.13	**0.91**

Table 5.3. Comparison of self-supervised AST with patch, frame, and VCR patch and frame

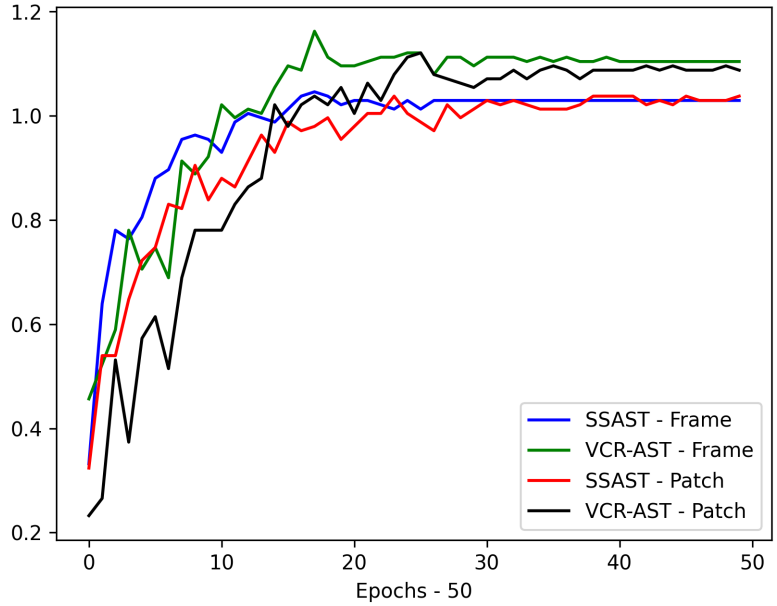


Figure 5.3. Normalized performance comparison between SSAST in Frame and Patch configuration, and VCR-AST in Frame and patch configuration

5.6 Ablation study

In this section, we perform an ablation study to understand the impact of various design choices on the performance of our proposed model. We investigate the following aspects: (1) the impact of different model sizes, (2) the impact of the number of masked patches during training (3) the impact of different sizes of batches of samples, and (4) the impact of a different dimension of embeddings of the projector during training. By understanding the contributions of these factors, we aim at providing insights into the robustness and generalizability of our model.

5.6.1 Impact of Model Size

We test two different configurations of the AST architecture to study how that influences the performances.

Model Size	Dataset	Loss	Avg Time per sample	# Parameters
Base	ESC50	21.5	0.045	125M
Small	ESC50	22.3	0.027	57M

Table 5.4. Ablation study on model size impact

- **Base model** is the default configuration, the transformer has 12 layers and 12 attention heads and embeddings of dimension 768, and adding our proposed method it reaches 125M parameters.
- **Small model** in this case transformer has 12 layers with 6 attention heads and embeddings dimension of 384, in total 57M parameters.

We compare the two configurations using the same parameters in the frame-based version and the results are reported in table 5.3.

The larger model obviously reaches better performance in general, but as you can

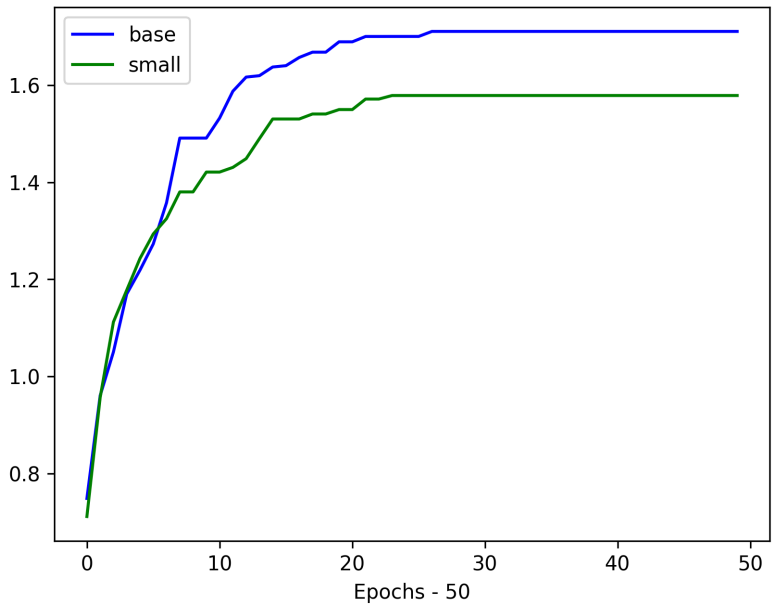


Figure 5.4. Normalized accuracy of Base size Model and Small size Model.

see, when trained with small data (1600 samples) it doesn't outperform the small model so much in training and reaches almost the same performances in fine-tuning. However, it still performs better and this demonstrates that scaling up the model has the potential to achieve better performances. Anyway, on small data, the small model can be a good choice since it employs half the time on average and almost half the space. Finally, **increasing learning rate in a fine-tuning stage can slightly improve small model performance, this impacts less the base model.**

5.6.2 Impact of Number of Masked Patches

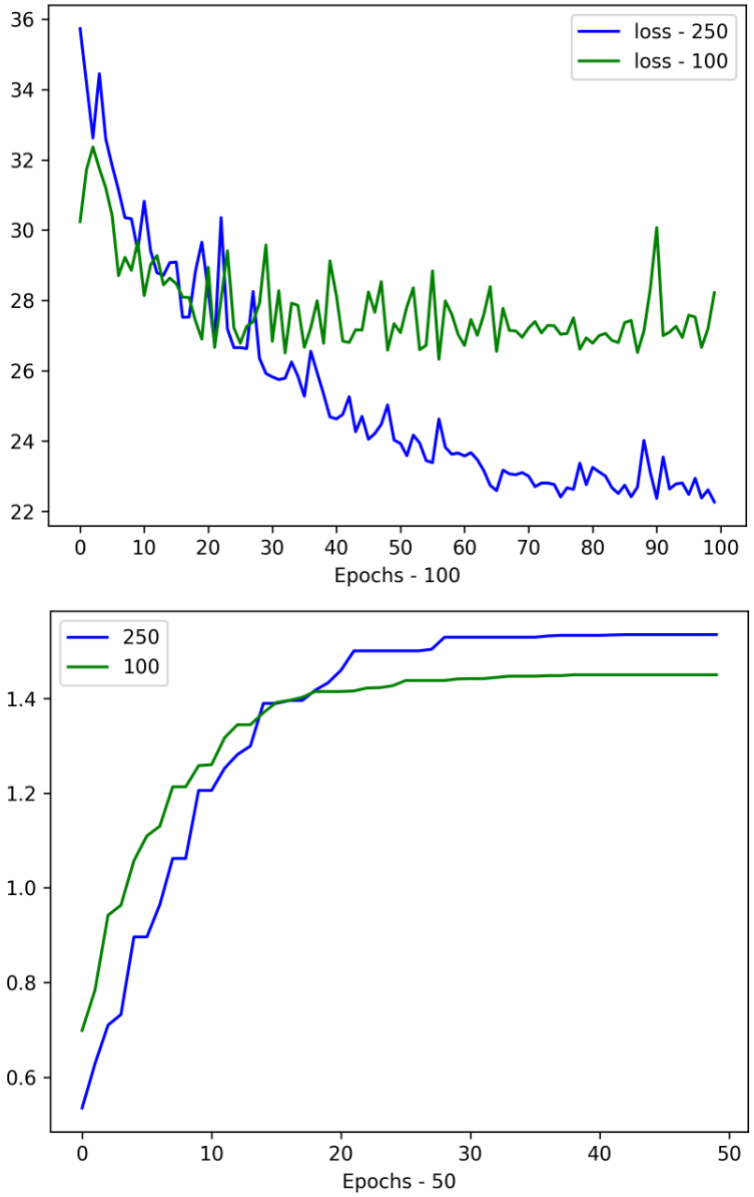


Figure 5.5. (1) Impact of number of masked patches on loss (2) Impact of number of masked patches on accuracy

Our analysis of Figure 5.5, revealed that a relatively low number of masked patches (100) resulted in the poorest performance across all downstream tasks. On the other hand, masking a higher number of patches (250) led to better performance on audio event classification tasks. While it may be tempting to mask even more patches in an effort to further improve performance, the reality is that our current hardware resources do not allow for such an approach due to the extended training time that would be required.

As a result, we were unable to explore the impact of masking more patches on downstream task performance. In Fine-tuning the model pretrained with 250 masked patches performs better but the difference is not incredibly evident. Anyway, we think that with a bigger amount of pretraining data, better results are achievable.

5.6.3 Impact of different Batch size

Pretraining the model with 6 and 12 as batch sizes, we can see how an higher batch size tends to reach higher results. The reason for this is that using a larger batch size allows the model to see more examples in each training iteration, which can help it generalize better to unseen data. When using a smaller batch size, the model may be too sensitive to the specific characteristics of the training examples in each batch, leading to overfitting. In particular, for the task of ESC50, where data is small, the risk of overfitting is high, and increasing batch size can be a solution.

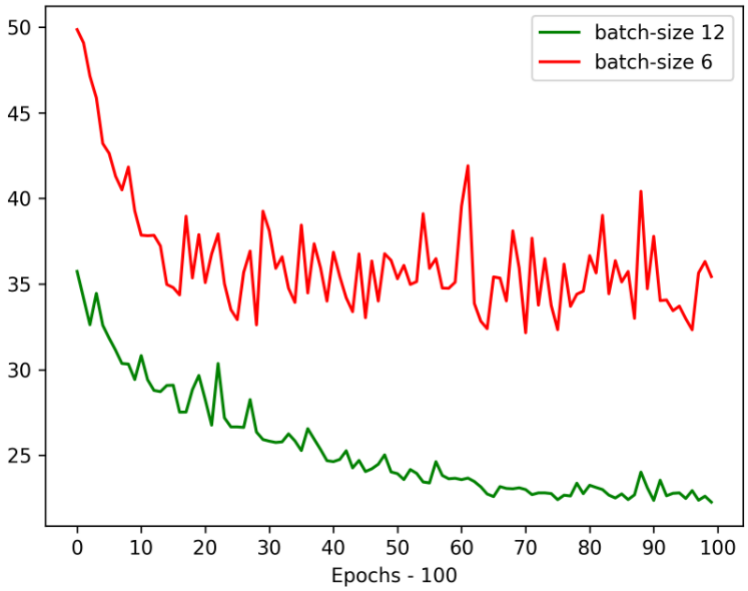


Figure 5.6. Impact of different Batch size on training loss and accuracy

5.6.4 Impact of different embeddings dimension of VCR Projector

We pretrain the model using two different sizes for the MLP head of the VCR Projector, standard is "4096-4096-4096" (it has three layers), and the alternative is "1024-1024-1024".
There are a couple of ways in which the projector dimension can affect the performance of our proposed method. First, if the dimension of the projector is too low, it may not be able to capture the complexity and diversity of the data, leading to poor performance. On the other hand, if the dimension of the projector is too high, it may overfit the data and lead to poor generalization of new, unseen data.
Additionally, the dimension of the projector can affect the computational complexity of the model, as a higher-dimension projector will require more computations

and resources. This can impact the speed and efficiency of the model, potentially leading to slower training times and less efficient use of resources.
In our testing cases, larger dimensions will allow the model to capture more information and improve performance (it may also increase the risk of overfitting if the model is too complex relative to the amount of training data available). Anyway, obviously it slows a lot the training and increases the size of the model in terms of space: the 4096 model is about 500MB while the 1024 version occupies about 350MB.

5.7 Experiments limitation

Anyway, our experiments on this work have been limited by our resources in several ways that may have impacted the accuracy and generalizability of our results. Firstly, we were unable to run as many trials or use as large of a dataset and multiple datasets as we would have liked due to constraints on our computing power. This is a common challenge in the field of machine learning, where the amount of data and the complexity of models can quickly exceed the capabilities of even the most powerful computers. As a result, we had to be selective in the data we used and the number of trials we were able to run, which may have introduced bias or limited the robustness of our findings. Secondly, we were unable to fully explore certain aspects of the approach outlined in the previous sections due to time and other limitations. While we were able to make progress on some of the core ideas, we were unable to fully test all of the hypotheses or delve into some of the more complex or nuanced aspects of the work. It is important to consider these limitations when interpreting the results of our experiment, as they may have influenced the outcomes we observed. Normally, AST is tested on a batch size of at least 48, 96 most of the time, In our case the most resources could use has been 12. In some experiments, a target length of 1024 is proved to provide better indications and we needed to lower it to 512. Some applications of methodologies similar to the proposed one use projector with MLP heads of 8192, we used 4096. Maybe the biggest limitation has been on datasets. We had resource limitations that didn't permit us to download entire datasets like AudioSet or FS50K. Anyway, those datasets require huge resources to make it possible to train the model in a certain time. Lastly, experiments of SSAST have been done on 4 x NVIDIA GTX Titan X for 800k iteration that took about 10 days, on datasets of more than 2 million samples, while in our case experiments have been done on a single GPU with limited time access for up to 20k iterations on a dataset of 2000 samples.

5.8 Future Works

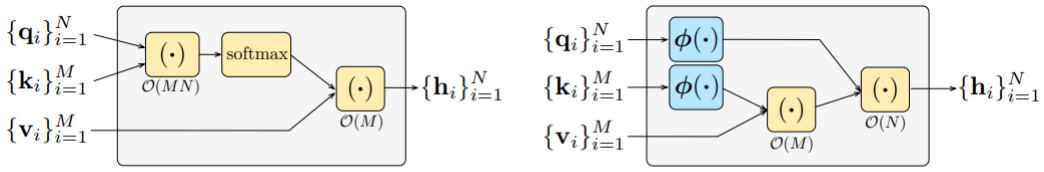
5.8.1 RFA - Random Feature Attention

Transformer architectures have been extremely successful in a wide range of sequence modeling tasks. One of the key elements that contributes to this success is the Attention mechanism, which captures the interactions between inputs, regardless of their relative positions. However, this also results in quadratic time

and memory costs, making the transformers computationally expensive for long sequences. To address this issue, the RFA (Random Feature Attention) [22] method has been proposed as a solution, as it scales linearly in terms of time and space, and provides practical benefits for both long and moderate-length sequences. RFA uses a kernel trick [23] to approximate the dot-then-exponentiate function:

$$\exp(x \cdot y) \approx \phi(x) \cdot \phi(y).$$

To further improve performance, the authors of RFA have also added an optional gating mechanism, which allows for the incorporation of recency bias while maintaining locality. This gating mechanism can be easily implemented as a replacement for the standard softmax attention, and only slightly increases the number of parameters (by less than 0.1%).



(a) Softmax attention. (b) Random feature attention.

Figure 5.7. Random Feature Attention

$$\exp\left(\frac{x \cdot y}{\sigma^2}\right) = \exp\left(\frac{||x||^2}{2\sigma^2} + \frac{||y||^2}{2\sigma^2}\right) = \exp\left(-\frac{||x-y||^2}{2\sigma^2}\right) \approx \exp\left(\frac{||x||^2}{2\sigma^2} + \frac{||y||^2}{2\sigma^2}\right) \varphi(x) \cdot \varphi(y)$$

The last line does not have any nonlinear interaction between $\varphi(x)$ and $\varphi(y)$, allowing for a linear time/space approximation to attention. For clarity, we assume the query and keys are unit vectors.

$$\text{attn}(q_t, k_i, v_i) = \sum_i \exp\left(\frac{q_t \cdot k_i}{\sigma^2}\right) \frac{\sum_j \exp(q_t \cdot k_j/\sigma^2) v_i}{\sum_j \exp(q_t \cdot k_j/\sigma^2)} \approx$$
$$\sum_i \varphi(q_t)^T \varphi(k_i) v_i \frac{\sum_j \varphi(q_t) \cdot \varphi(k_j)}{\sum_j \exp(q_t \cdot k_j/\sigma^2)} = \varphi(q_t)^T \sum_i \varphi(k_i) \otimes v_i \frac{\sum_j \varphi(q_t) \cdot \varphi(k_j)}{\sum_j \exp(q_t \cdot k_j/\sigma^2)} =$$
$$\text{RFA}(q_t, k_i, v_i).$$

RFA utilizes a kernel method perspective to analyze softmax attention and substitutes it with random feature methods. The inclusion of a gating mechanism allows for easy learning with recency bias. **The linear time and space complexity of RFA in relation to sequence length makes it a suitable alternative to softmax attention in transformer models.**

5.8.2 AST on Quaternion domain

Neural networks have shown great potential in a wide range of applications, and recent work has explored the use of neural networks on the quaternion domain for various tasks. Now we focus on the potential of neural networks in the quaternion domain, particularly for transformers and audio classification.
Quaternions are a type of mathematical object that extend complex numbers to

four dimensions and can represent rotations in 3D space more efficiently than traditional methods such as Euler angles or rotation matrices.
One area where quaternion neural networks [24] have shown promise is in natural language processing tasks, such as machine translation. Quaternion transformers have been proposed as a variant of the popular transformer architecture that operates on quaternion-valued inputs and weights. These quaternion transformers have been shown to achieve competitive results on several benchmarks, including the WMT 2014 English-to-German translation task and the IWSLT 2014 German-to-English translation task.
In the field of audio classification, quaternion neural networks have also been explored as a way to improve performance. For example, quaternion convolutional neural networks (CNNs) have been applied to tasks such as music classification and speaker identification. These networks can learn rotation-invariant features from the audio data and have been shown to outperform their real-valued counterparts in some cases. Overall, the potential of quaternion neural networks is an active area of research and we will likely see more applications of these networks in the future.
Another area where quaternion neural networks have demonstrated potential is in audio classification tasks. Audio signals can be represented as quaternion-valued time series, and quaternion convolutional neural networks have been used to classify such signals. These quaternion CNNs have been shown to outperform their real-valued counterparts on several audio classification tasks, including speaker identification and speech recognition.
Overall, the potential of neural networks on the quaternion domain is an active area of research, and these early results suggest that quaternion neural networks may be a promising approach for a range of tasks. Further research is needed to fully understand the capabilities and limitations of quaternion neural networks and to identify the most effective ways to apply them.

Chapter 6

Conclusions

This work's objective was to introduce a new self-supervised approach to AST - Audio Spectogram Transformer via Variance-Invariance-Covariance regularization. AST is one of the most important works on audio scene analysis as a supervised method, it introduces a purely self-attention convolution-free way that is pretrained on Imagenet needs less audio data, and thanks to transfer learning outperforms CNNs. The main problem with this solution is the amount of data and time needed to achieve these results. The best solution from this point of view is to adopt self-supervised methods, and in this sense, SSAST is the state-of-the-art method right now on a lot of downstream tasks. Our proposed method is also self-supervised but introduces some advantages: improvements in generalization, data efficiency, and flexibility. Even if this is not a brand new methodology, a similar one is only used for self-supervising a Resnet and most of the works are on image datasets and tasks. Instead, to the best of our knowledge, this is the first application of a self-supervised method using Variance-Invariance-Covariance regularization on a Transformer architecture, in particular: AST. In addition, our is one of the first applications of the method on audio datasets, using audio files converted into spectrograms like images. The greatest problem with this type of datasets is the amount of data. Most famous image datasets contain millions of samples, while audio datasets are significantly smaller, in addition, the greatest ones are not labeled. Without applying transfer learning, self-supervised learning is the only solution to train a model with audio datasets. The problem now is that using small datasets the model tends to overfit: to avoid overfitting we tried different configurations of the model using different strategies. Increasing the projector embeddings has proved to be a good solution since the model can extract the most feature possible, but implies considerable resources in terms of Hardware and time. Another key to help the model generalize better the data is to increase the batch size, in this way the model can see more samples per iteration. The same disadvantage of this approach is the resources needed. Finally, the model has proved to enjoy challenges and reacts better when data is masked more when masking data with 100 masked patches, the model overfits and stops improving very fast. Contrarily, with a higher number of masked patches (250) the model continues learning. Unfortunately, during our experiments, we have been limited by the resources available, and we chose to limit some parameters to make it possible to train the model for a certain amount of iteration in a reasonable time. We think the method has potential, and results seem to be promising on a Large scale.

Acknowledgments

First of all, I want to thank the University for giving me the opportunity of achieving this dream. Secondly, I want to thank my advisor, Prof. Comminiello, it's been an honor to collaborate with you through these months. I've been inspired by your passion for this field of study and I feel fortunate for having found an advisor that has helped and followed me during the thesis' period, so, Thank you. Then, I want to thank all my friends and colleagues for having collaborated, studied, and inspired each other through these long and stressful years, again I feel lucky, thank you. Finally, I want to thank my family: my Grandmother (I know you're reading, I did it, I love you), my Mom, and my Brother, since I started having consciousness my life's goal has always been your happiness, you inspired and helped me during difficult periods and gave me a strong attitude toward never give up and success whatever it takes. I hope I've made you proud 'till now, our future will be shiny, I swear. Last but not least, I want to thank my girlfriend, Martina, you've been by my side on this path. When I was sad, stressed and unbearable (really unbearable) day and night, you supported me and never left me alone when everyone wouldn't. I really know It's not been easy, you've been strong and inspiring. Again, whatever it takes, I'll try to be the best version of myself for you, I love you. In conclusion, Thank you all, everyone is part of this goal and part of what I am now.

P.S: For my future self, Always remember all the sacrifices you did and never regret them, respect yourself whatever you've become because you always did your best and I'm sure I will be proud of you. Never forget Where and Why you started, it's always been your fuel and it will ever be.

Bibliography

[1] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio Spectrogram Transformer," in *Proc. Interspeech 2021*, pp. 571–575, 2021.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[3] A. Bardes, J. Ponce, and Y. LeCun, "Vicreg: Variance-invariance-covariance regularization for self-supervised learning," in *ICLR*, 2022.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *ICLR*, 2021.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[6] S. Bhojanapalli, A. Chakrabarti, D. Glasner, D. Li, T. Unterthiner, and A. Veit, "Understanding robustness of transformers for image classification," 2021.

[7] R. Shao, Z. Shi, J. Yi, P.-Y. Chen, and C.-J. Hsieh, "On the adversarial robustness of vision transformers," 2021.

[8] K. Mahmood, R. Mahmood, and M. van Dijk, "On the robustness of vision transformers to adversarial examples," 2021.

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

[10] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pp. 1015–1018, ACM Press.

[11] P. Warden, "Speech commands: A public dataset for single-word speech recognition.," 2017.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[14] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," 2020.

[15] S. Liu, A. Mallol-Ragolta, E. Parada-Cabeleiro, K. Qian, X. Jing, A. Kathan, B. Hu, and B. W. Schuller, "Audio self-supervised learning: A survey," 2022.

[16] Y. Gong, C.-I. J. Lai, Y.-A. Chung, and J. Glass, "Ssast: Self-supervised audio spectrogram transformer," 2021.

[17] L. Wang and A. v. d. Oord, "Multi-format contrastive learning of audio representations," 2021.

[18] A. Baade, P. Peng, and D. Harwath, "Mae-ast: Masked autoencoding audio spectrogram transformer,"

[19] K. Chen, X. Du, B. Zhu, Z. Ma, T. Berg-Kirkpatrick, and S. Dubnov, "Hts-at: A hierarchical token-semantic audio transformer for sound classification and detection," in *ICASSP 2022*.

[20] Y. Gong, C.-I. Lai, Y.-A. Chung, and J. Glass, "Ssast: Self-supervised audio spectrogram transformer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 10699–10709, 2022.

[21] S. Atito, M. Awais, W. Wang, M. D. Plumbley, and J. Kittler, "Asit: Audio spectrogram vision transformer for general audio representation," 2022.

[22] H. Peng, N. Pappas, D. Yogatama, R. Schwartz, N. A. Smith, and L. Kong, "Random feature attention," 2021.

[23] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The Annals of Statistics*, vol. 36, jun 2008.

[24] T. Parcollet, M. Morchid, and G. Linarès, "A survey on quaternion neural networks," 2020.