Master in Sound and Music Computing

Universitat Pompeu Fabra

Applying Computer Vision Foundation Models to Audio Source Separation

Benjamin Olsen

Supervisor: Marius Miron

Co-Supervisor: Genís Plaja

August 2023

Master in Sound and Music Computing

Universitat Pompeu Fabra

Applying Computer Vision Foundation Models to Audio Source Separation

Benjamin Olsen

Supervisor: Marius Miron

Co-Supervisor: Genís Plaja

August 2023

Contents

Abstract

In this paper we attempt to apply foundational models in computer vision to the problem of audio source separation. Focusing on separating a foreground sound from a mixture, we attempt to leverage the strong feature learning capabilities of these models to perform image segmentation on audio spectrograms. Using an augmented version of the FSD50k dataset, we first asses the new models DINOv2 and the SAM (Segment Anything Model) in a zero-shot transfer learning scenario, then under different fine-tuning methods. We hope to find state-of-the-art performance.

Keywords: Audio Source Separation; Transfer Learning; Foundational Models; Image Segmentation; Deep Learning

Chapter 1

Introduction

The aim of this thesis is to explore the possibilities of new foundation models in computer vision applied to the domain of audio source separation.

In the earlier days of machine learning, models were typically trained from scratch for each new task, which requires large amounts of task-specific data and computational resources. This began to change in the late 2010s when transfer learning, the act of applying a model trained for one task to a distinct but related task, started to become possible.[1][2][3]

Around the same time, models began to grow significantly in size as researchers found that model performance often improved with increasing the amount of training data, model depth (number of layers), and/or width (number of channels, or neurons). [4][5][6] The latest manifestation of this progression is that certain models are being trained with a more general purpose on datasets which are equally general, and as a result, vast. We call these new types of models Foundational Models. The term can be applied generally to any number of AI systems which are trained on a very large amount of data, usually using self-supervision at scale, and often boasting creative training data pipelines as their main innovation, rather than a model architectural improvement. Further discussion of the topic will be found in the State of the Art chapter.

1.1 Motivation

Deep learning is a very cross-disciplinary field, where discoveries in one field of study can often have implications in many others. Case in point, the original U-Net architecture[7] was originally developed for the specific task of biomedical image segmentation, but was quickly adapted by the research community to tackle satellite and aerial image segmentation [8], super resolution imaging[9], image-to-image translation[9], denoising and image restoration[10], and of course music source separation with Open-Unmix[11] and Wave-U-Net[12].

The models which we adapt in this work also have shown cutting edge performance in image segmentation, and so the idea was born to follow in the same vein as the above research to apply them toward source separation.

Intuitively, the task of image segmentation is bears similarities to audio source separation - in the sense that the task can be reduced to differentiating one "thing" from another "thing" which is represented in some data. That data can be visual or sonic or otherwise, but as long as they can be represented digitally, then we can turn to Deep Learning to find complex patterns therein, specifically in this work we deal with Vision Transformers.

1.2 Objectives

The objective of this thesis is to assess the performance of some of these new foundation models in computer vision in a zero-shot, few-shot, and more-shot transfer learning scenario for audio source separation.

1.3 Structure of the Report

First, we will provide an overview of the field of audio source separation in the State of the Art chapter, including an introduction to the models DINOv2 and SAM that will be the backbone that this work attempts to add to. This is followed by a detailed discussion of how these models are adapted to an audio source separation

task in the Architecture chapter. The data we will use to train, test, and verify the resulting source separation models is then discussed in the Datasets chapter. The training and verification procedure is detailed in the Experiment Setup chapter, the results are then presented in the following chapter, and the paper is finished off by a discussion of the whole work.

Chapter 2

State of the Art

This section is to give an overview of the field of source separation, emphasizing its application to audio, as well as touching upon the emerging area of foundation models.

As it is a very rich field of research, this is necessarily an incomplete summary of it, as to do a more exhaustive one would require a book length review. The structure of it reflects my own personal study path - starting in ignorance, and hopefully ending in some sort of mastery. In that vein, it contains perhaps more historical and introductory material than is usual for a State of the Art, but I've tried to not make it too silly. I've tried to keep the structure of this work narrative in a conceptual level so that one part follows from the preceding. Often, this means chronological with the emerging research, but sometimes not.

2.1 A brief overview of source separation

A familiar way to introduce the problem of source separation is in the context of speech. The "cocktail party effect," first described by Cherry in 1953[13], is the ability of the audio cognitive system to focus on a single conversation out of many, as would often happen during a 1950s cocktail party. Extend this idea to general audio environments, and you have defined Audio Source Separation. A. S. Bregman

described the psychoacoustic function of taking sensory input and deriving a useful representation of reality from it in his 1990 book Auditory Scene Analysis[14]. Attempts to perform the same with computational methods followed the deepening understanding of these capabilities. The first attempts focused on extracting semantic features from mixed sources [15], but soon extended to the general problem of deconstructing an audio scene into its constituent waveforms.

Put mathematically, the question assumes the existence of n independent signals $s_1(t), ..., s_n(t)$ and the observation of some number mixtures $x_1(t), ..., x_m(t)$, with these mixtures being linear and instantaneous, i.e. $x_i(t) = \sum_{j=1}^{m} \alpha_j s_j(t)$ for each $i \in [1, ..., n]$, $\alpha_j \in \mathbb{R}$. The goal of source separation is to estimate each $s_j(t)$ given each observation $x_i(t)$, and perhaps some a priori information about the set of $s_j(t)$.

2.2 Filtering

One way of viewing the problem of source separation is to view it as an act of filtering. Using the notation from above, a filtering operation is simply a function $\mathbf{F}_j$, which applied to a mixture signal $\mathbf{F}_j(x_i(t)) = \hat{s}_j(t)$ produces an estimation of a source signal. For example, a model can learn to separate vocals from a mixture by applying a learned time varying filter. The performance of the separation can be then quantified by comparing a known source $s_j(t)$ and its estimation $\hat{s}_j(t)$.

2.3 Blind source separation

The problem of recovering signals from mixtures with only partial knowledge of the original signals and of the mixing process is known as blind source separation, or BSS. The simplest BSS model assumes the existence of n independent signals $s_1(t), ..., s_n(t)$ and the observation of as many mixtures $x_1(t), ..., x_n(t)$. The mixing equation is then $x(t) = As(t)$ where $s(t) = [s_1(t), ..., s_n(t)]^T$ is a $n \times 1$ column vector of the source signals. The vector $x(t)$ collects similarly the n observed signals, and the square $n \times n$ "mixing matrix" A contains the mixing coefficients. [16]

Independent Component Analysis (ICA) is an early example of this attempt to

separate a multivariate signal into its components. It attempts to decompose a mixture into statistically independent signals, according to a specific non linear measure of independence. With only access to samples of a mixture of distributions, there are many ways to estimate the level of independence [17][18][19]

There are different criteria for independence and various optimization algorithms that fit within the family of ICA, but each of them require the mixture to be "over-determined", meaning that there are at least as many mixtures as the number of sources it seeks to identify. The problem of separating sound sources from a single mixed input quickly became a major focus in the field of source separation.[20], in his paper One Microphone Source Separation, describes a method that leverages the fact that real life audio comes from a specific distribution instead of any possible one, and so by modeling waveforms' distribution, one can extract a likely combination of sources. The approach, which he calls refiltering, uses hidden markov models (HMM) to claim ownership of each time-frequency bin of the amplitude spectrum, creating binary masks of it for each source, which can be used to filter the mixture to recover the sources.

2.4 Image segmentation

This leads us nicely into the realm of image segmentation, which is precisely where we will end up later in this thesis. In Bach and Jordan's 2005 paper Blind one-microphone speech separation: A spectral learning approach, the researchers use features adapted from psychoacoustics to train an image segmenter specific to speech[21]. Some of the features, for example, are "common fate" cues ("elements that exhibit the same time variation are likely to belong to the same source"); timbre; and spectral clustering. These features are, however, hard-coded instead of learned directly from the training data, which is a limitation.

The crux of applying image segmentation in source separation is that it converts the separation problem into one of classification: Yilmaz and Rickard noted in 2004 that if the ideal mask is known, then applying a binary mask with coefficients in

$\{0, 1\}$ to the amplitude spectrogram, followed by using the input phase to reconstruct the waveform from each source image yields convincing results.[22] Reddy and Raj refined this approach in 2007 by using a soft mask, using real coefficients in the interval $[0, 1]$, which is appropriate in many real-world cases where sources' frequencies overlap.[23]

2.5 Matrix Decomposition

Similar to this is Non-negative Matrix Factorization (NMF), initiated by Paatero & Tapper in the 1990s[24] but carried on by Lee and Seung in 2001, who coined the term itself[25]. The decomposition is into two matrices: one a set of bases, and the other of weights. The technique was applied successfully to a variety of problems, but Smaragdis pointed out that an extension of NMF applied to time series could be useful for source separation for single channel inputs.[26] Using the fact that power spectra are strictly non-negative, you can approximate it as a product of two non-negative matrices: the bases effectively a dictionary of base spectral patterns corresponding to sources, and the other the time weighting of the dictionary entries. Sources with complex spectral patterns, however, end up spanning more than one base. "In the case of music signals, each component usually represents a musically meaningful entity or parts of it, so that different entities are represented with different components. The entities can be for example the sounds produced by a percussive instrument or all equal-pitched notes of a pitched musical instrument. This representation is enabled by the spectral structure of musical sounds, which is usually rather static over time compared to speech signals, for example."[27] NMF can compute the weights for a given amplitude spectrum over the set of sources, and separation is done through a variety of manners. One, as described in Virtanen[27], for example, is by factoring the spectrogram of the input signal into its bases, and then grouping them into the individual sources to reconstruct each source individually by choosing its correct set of bases and weights. An important assumption that is made, here, is that sound sources add linearly in the time domain when they are mixed.

2.6 Supervised source separation

One way of dealing with the issue of complex sound is through using supervised data: if recordings of individual sources are available, then the model can determine the set of bases for each source during training, which greatly simplifies the matter.

2.7 Source separation metrics

Supervised data also allows for objective metrics to assess a source separation system's performance. Thanks to the work of Schoben et .[28], and Vincent et al. [29], there are a few standard metrics for relative amounts of distortion, interference and artifacts, but the two main metrics are Signal to Distortion Radio (SDR) and Signal to Interference Ratio (SIR), with the first measuring the quality of the reconstruction, and the second quantifying the amount of leakage from other sources to a separated source.

2.8 Evaluation campaigns

Vincent et al organized the first evaluation campaign for audio source separation, where thirteen research teams submitted algorithms to set against each other using a dataset of 10 second audio clips.[30] The most recent SiSEC took place in 2018[31]. In the 2007 campaign, the main focus of the field was with NMF or ICA; whereas in 2018 deep learning techniques had taken over.

2.9 Deep Neural Nets

The SiSec 2018 campaign had 30 systems submitted, and all of them were deep learning systems. The state of the art, from here on, consists of many such systems, and whenever a new one is discovered, the likely outcome is on its name ending with "-Net". For example, Uhlich et al. 2015 used a deep neural network to extract instruments from a mixture only with knowledge of the instrument types that are present in the mixture.[32] The DNN was trained on data from a database of solo

instrument performances.

2.10 The problem with the STFT

As of 2018, most audio source separation methods rely on spectrogram representations of the audio signals. It is a convenient representation for a number of reasons: for example, it allows for direct access to time and frequency components; there are fast algorithms to compute it (STFT and its inverse); the resulting two-dimensional data can borrow from work in image processing to apply it directly to audio. However, there are shortcomings as well. Firstly, the STFT itself is dependent on various parameters which affect the temporal and frequency resolution of the spectrogram, namely the window type, the window size, the FFT size, and hop size. The choice of these parameters is dependent on both the nature of the sound and the intended goal (e.g. frequency or time discrimination), therefore subtle changes in them can have profound effects on the quality of analysis and synthesis, and therefore the quality of separation and its metrics. It is not common to see reasoning for selection of the STFT parameters in papers, though they might warrant a mention in a section mentioning how certain parameters are tuned. A solution to this, used by Venkataramani and Smaragdis in their work investigating how to do end-to-end source separation, was to have the transform from the waveform delegated to its own neural network: an auto-encoder neural network that can act as an equivalent to short-time front-end transforms.[33] This approach tries to kill two birds with one stone: instead of using the complex valued Discrete Fourier Transform (DFT), it uses the real valued Discrete Cosine Transform (DCT).

2.11 The problem of phase

How to treat phase information in separation tasks is a complicated issue. On one hand, Human hearing is relatively indifferent to phase, and many of the matrix decomposition methods of source separation rely on the assumption that sounds add linearly in the time domain. But, in reality, sounds interfere with each other when mixed. In other words, phase is crucial. The output of the STFT is a complex num-

ber, and while most of the information that is used to distinguish between sounds lies in the amplitude of the complex value, the phase is crucial to synthesize realistic audio. The phase is also very unpredictable for real world sources. Below, figure 1 is the magnitude and phase spectrogram of a recording of a cello, and figure 2 that of a recording of a human voice singing. As you can see, the phase spectrum simply looks 'noisier' than the magnitude spectrum, which is reflected in the difficulty in selecting characteristic features of the phase for discrimination purposes. A logical question would therefore be: what about deep learning models? Could they learn something that the human eye cannot in the data? The performance of state-of-the-art source separation models have improved to the point that the noisy phase has now become a limiting factor. In fact, when measuring performance through SNR, beyond a certain point, improving the magnitude estimate may lead to worse results when combining it with a noisy phase - for example if the noisy phase is opposite to the correct phase, it simply cancels the contribution of the frequency.[34]

Some researchers have taken phase into account in various ways, to varying degrees of success [35],[36][34]

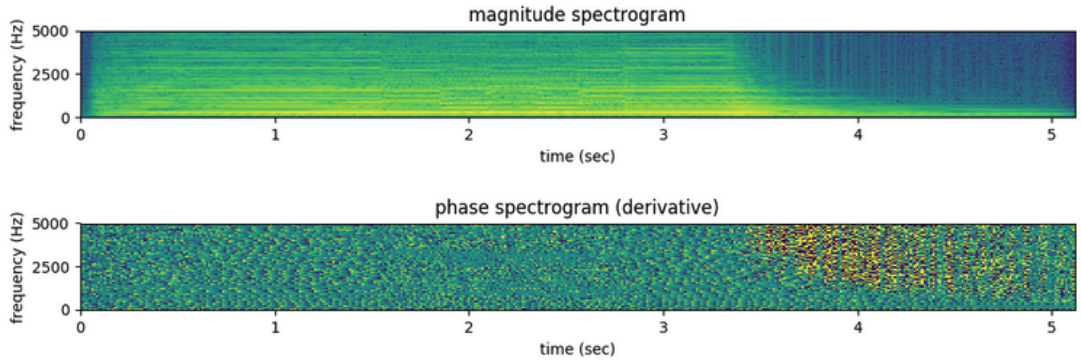

Figure 1: A bowed cello phrase (source: author)

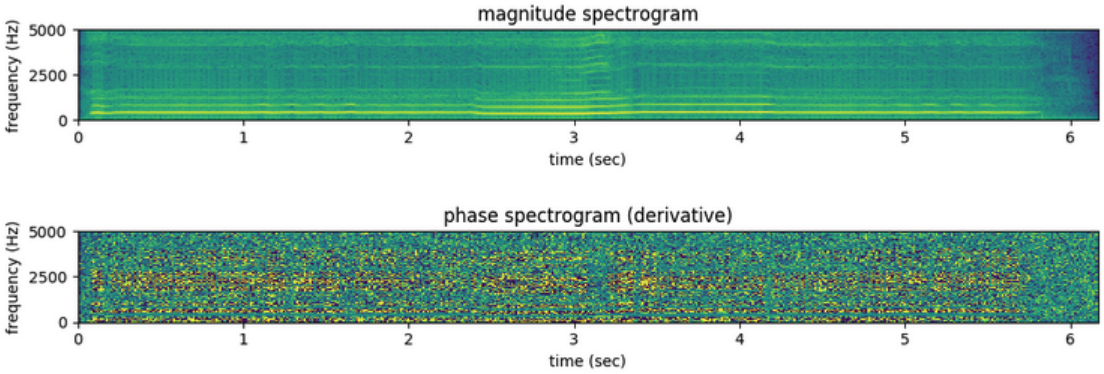

Figure 2: A human-sung phrase (source: author)

2.12 Denoising convolutional auto-encoders

An interesting application of neural networks is the case of the autoencoder, which finds an efficient coding (and hopefully decoding) of its input data. Effectively, it reduces the dimensionality of input data in the latent space of the model One way of seeing this reduction of dimensionality is that it is a sort of 'widening' of a field of view. It is an important idea in deep learning architectures which seek to learn features of data on both a small and large scale. (Akin to the evolution from Recurrent Neural Networks (RNNs) to Long-Short Term Memory (LSTM) networks to preserve temporal information in sequential data.) Coming back to audio, in the special case of singing voice extraction, convolutional encoder-decoder architectures have been explored. Chandna, Miron, Janer, & Gómez' paper Monaural Audio Source Separation Using Deep Convolutional Neural Networks from 2017 uses such a framework, and was competitive although not best-in-class.[37] Similarly, Grais & Plumbley did

so using a conditional denoising auto-encoder trained on each source.[38]

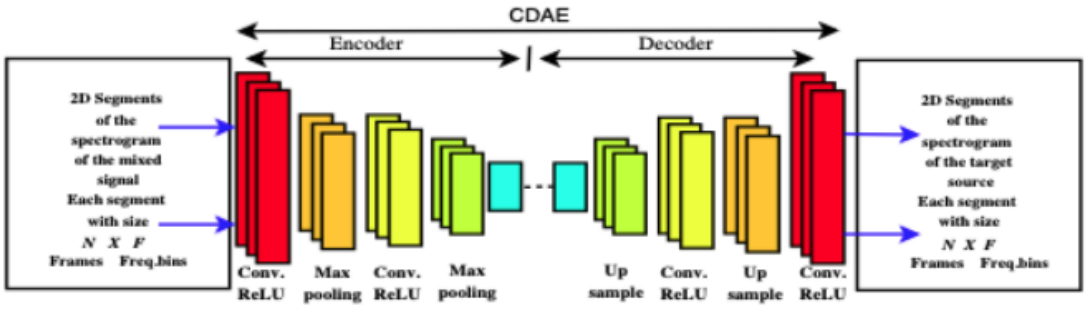

Figure 3: architecture of one CDAE, which separates one source, source: Grais & Plumbley (2017)[38]

2.13 Waveform based source separation

One solution to the limitations of using spectrograms in source separation is to bypass them completely, and operate on the raw audio waveform level. Waveforms are high-dimensional and very variable, meaning they're quite difficult to model in conventional neural network architectures. Early applications of deep learning to waveforms include a 2017 paper from the Google DeepMind London group of van den Oord et al, which presents WaveNet - a deep neural network for generating raw audio.[39] It operates on a serial, sample-by-sample prediction level, and can be trained to model arbitrary sound sources with excellent results. One of the problems with waveform based generative models is modeling long-range temporal dependencies, which they address by using dilated causal convolutions. The principle of dilated convolutions is a way of applying a convolutive filter over a larger area than its length by perforating the input rather than pooling. In the images below we see a comparison between causal convolution layers and dilated causal convolution layers. The dilation refers to the fact that only certain values are taken from each subsequent layer, the rest are set to zero - this allows for a widening of the field of view in each layer without the additional computational cost of pooling.

A more relevant example is with U-Net, a network developed by Ronneberger et al. for biomedical image segmentation in 2015.[7] It uses deep convolutional networks, which have been at the state of the art in many visual recognition tasks, starting with

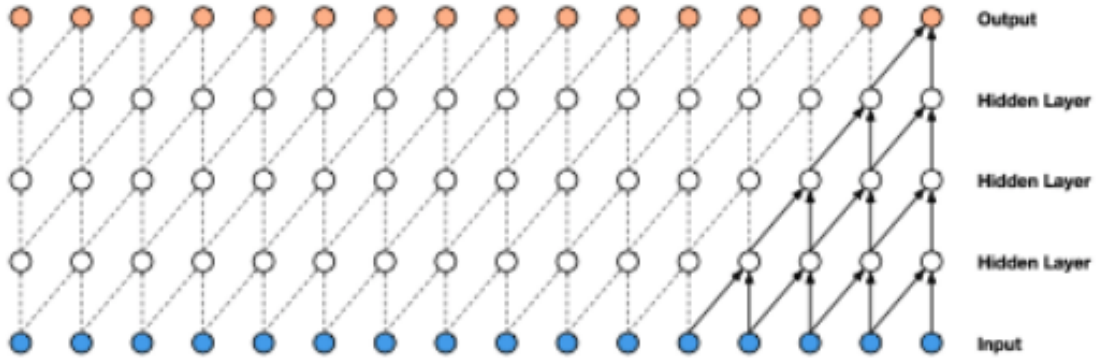

Figure 4: Visualization of a stack of causal convolutional layers, from van den Oord et al. 2017[39]

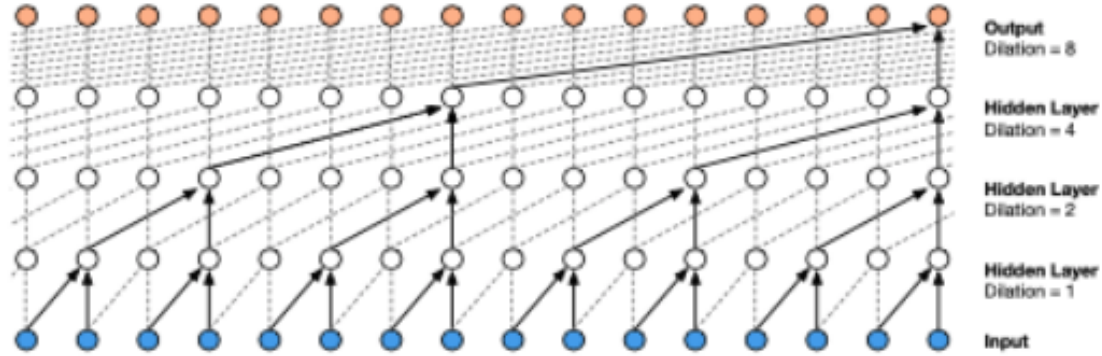

Figure 5: Visualization of a stack of *dilated* causal convolutional layers, from van den Oord et al. 2017[39]

the seminal Very Deep Convolutional Networks for Large-Scale Image Recognition by Simonyan and Zisserman in 2014 which used 16-19 layers of convolution to blow everybody's mind wide open.[40] U-Net likewise uses convolutional layers, but in a contracting path and a symmetric expanding path, through which the size of the feature map contracts through max pooling and expands through convolution. The stated purpose of the contracting path is to capture context, and the expanding path for precise localization of features. The intuitive picture here is that if you reduce the resolution of an image, you can see the gross features more easily, and conversely with higher resolution you're more attuned to finer detail. The performance of U-Net for medical image segmentation exceeded the best method until then (a sliding-window convolutional network), as well as being computationally fast: a 512x512 image can be segmented in less than a second on a GPU.

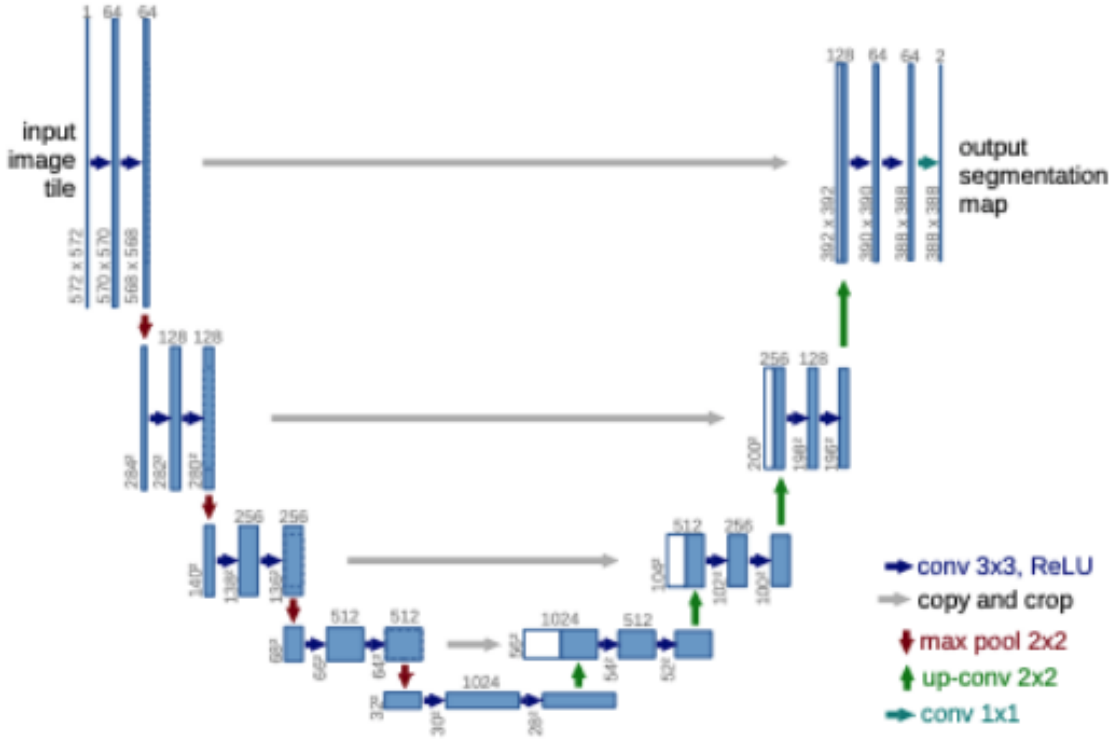

Figure 6: Visualization of the U-Net architecture. From Ronneberger et al. (2015) [7]

U-Net was shortly thereafter adapted to source separation using the magnitude spectrogram by Jansson et al. (2017), achieving state-of-the-art performance. [41]

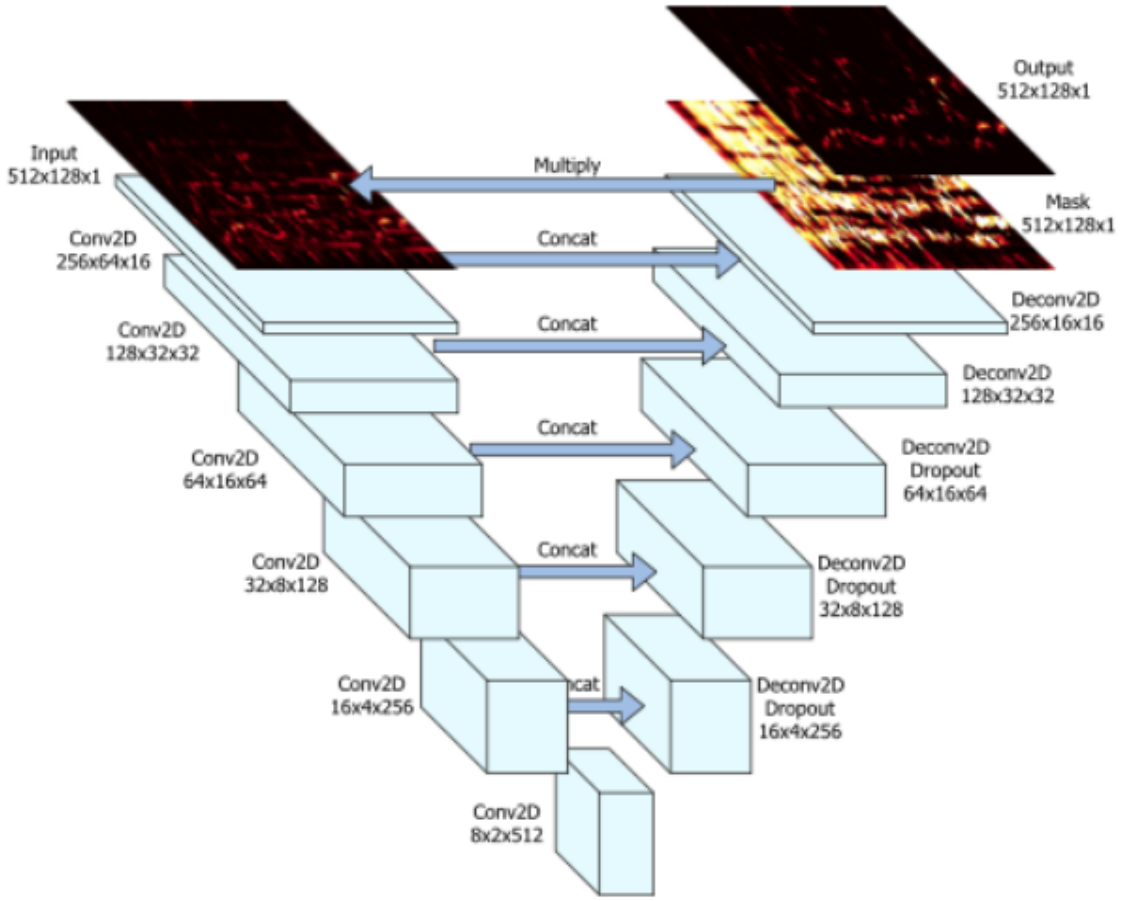

Figure 7: U-net for source separation architecture, from Jansson et al. (2017)[41]

This is done in a similar but different way by Wave-U-Net, from Stoller, Ewert, and Dixon in 2018.[12] It adapts the U-Net architecture to the 1-d time domain and deals directly with the source separation problem using different layers of down, and then up-, sampling to generate activations to different scales of features in the waveform. It computes an increasing number of coarser, higher-level features on the downsampling side, which are then concatenated with the later features on the upsampling side, making multi-scale features which are used for making predictions.

Lluís, Pons, and Serra posed the question End-to-end music source separation: is it possible in the waveform domain? in their paper of the same name, and answered it in the affirmative.[42] They present a model based on wavenet, borrowing an idea from Rethage et al. [39], modifying it to be non-causal and parallelizable, and therefore faster. Although their results were not as good (approx. 1.5dB worse) than the best spectrogram-based models from the 2018 SiSEC campaign, they succeeded

in showing the feasibility of end-to-end source separation.

2.14 Generative modeling in a nutshell: density modeling for data synthesis

Let's take a step back to look at generative modeling through a wider lens to set the stage for the last section of this where we take a look at diffusion models. Generative modeling can be seen simply as acting on probability distributions. The end goal is to generate a something that, under a certain measure of likelihood, is likely to belong to a known set of somethings which can be described statistically by a probability distribution. The act of training is the act of learning the probability distribution of a given dataset, and the act of generation is simply sampling from that learned probability distribution. The process of training is accomplished through minimizing the "distance" between the given distribution and the modeled one. There are plenty of ways to do this, but the benefit of deep learning models is that they can effectively learn very complex distributions for a variety of data.

There are various ways of measuring likelihood which a generative system attempts to maximize during its approximation of a given distribution using various divergence metrics. Some common examples are Kullback-Leibler (KL divergence) or Jensen-Shannon divergence. Diffusion models, take advantage of the properties of this process by training a system to reverse it.

Once you've learned the distribution, one of the ways to generate a sample is to feed some random noise into the system and convert it to something that looks like something that belongs to that distribution. GANs look at some latent noise vector, pass it to a generator, out pops an image; similarly for VAEs, and Flow-based models. They do it in a single step, more or less, where you give the neural network some noise and it gives you a sample, and that might be difficult to analyze.

2.15 Foundation Models

Now, let's switch gears a bit and briefly discuss the general idea of Foundation Models, as this will situate this historical retrospective in this section back into the present work. Foundation models refer to large-scale machine learning models that serve as a base, or "foundation", upon which more specialized models or applications can be built. They've become especially prominent with the advent of massive pre-trained models in natural language processing (NLP), like OpenAI's GPT[43] series and Google's BERT[1]. The key ideas and characteristics of foundation models are as follows:

Pre-training and Fine-tuning

Foundation models are often pre-trained on vast amounts of data (often public or general data) to learn generic patterns. Once pre-trained, they can be fine-tuned on specific tasks or datasets to adapt them to more specialized applications.

Broad Applicability

A single foundation model can be fine-tuned for a wide range of tasks. For instance, a model like GPT-3 can be used for chatbots, text generation, code writing, and more, with appropriate fine-tuning.

Scale

Foundation models are typically large, often comprising billions or even trillions of parameters. The scale of these models contributes to their capacity to capture intricate patterns in data.

Data-driven

The capability of these models emerges primarily from the vast amounts of data they're trained on, rather than domain-specific knowledge hard-coded into them. This is brought about by large scale data curation pipelines (as we'll see with the DINOv2 and SAM projects), where properties emerge from self-supervised learning

that does not appear in a supervised manner (as with Vision Transformers, which we'll discuss more below).

Economies of Scale

Training foundation models requires substantial computational resources, often making it prohibitive for most organizations. However, once trained, they can be fine-tuned and adapted at a fraction of the original computational cost, making them accessible to a broader audience.

Challenges and Concerns

Given the power of these models, there are (as with all technologies), substantial concerns and challenges to address concerning them:

Bias and Fairness: Given that they're trained on vast datasets, foundation models can inadvertently learn and propagate societal biases present in the data.[44][45][46]

Interpretability: Due to their size and complexity, these models can sometimes act as "black boxes", making it hard to understand why they make certain decisions.[47]

Over-reliance: There's a risk of the community overly relying on a few popular foundation models, potentially stifling innovation or leading to monocultures in machine learning.[48]

Collaborative Development: Given the significance and impact of foundation models, there's a growing interest in developing them collaboratively, involving a broader set of stakeholders, including academia, industry, and civil society.[49]

In summary, foundation models are like "base layers" in machine learning, offering a starting point that can be tailored to a myriad of specific applications. While they bring about impressive capabilities and efficiencies, they also come with challenges and responsibilities, especially concerning their ethical and societal implications. We'll get more into this, hopefully, in the discussion at the end of this work.

2.15.1 Intro to Vision Transformers

Transformers are a type of neural network architecture introduced in a paper called "Attention is All You Need" in 2017[4].

They were initially designed for handling sequence data, and their primary advantage is the ability to capture long-range dependencies therein, which are hard for recurrent neural networks (RNNs) to handle, for example. This property makes them particularly useful for tasks like machine translation, where the understanding of a word can depend on another word that appears much earlier or later in the sentence.

At the core of the transformer model is the concept of attention, and in particular, self-attention or scaled dot-product attention. Attention mechanisms let the model weigh the importance of a token's relationship to other tokens in the sequence, which means they give the model a way to focus more on certain tokens when processing a sequence of tokens.

ViTs were introduced in 2020 by Google[50], applying a simple method of serializing image data to a transformer backbone by turning an image into a series of patches paired with a positional encoding, which would retain the information of where in the 2-d space each of the image patches are found. The transformer then finds dependencies within the image data on varying scales. They have exhibited high performance in this respect, often surpassing the revolutionary convolutional neural networks in image classification [50], object detection [51], image generation [52], video understanding [53], fine-grained image recognition [54], transfer and few-shot learning [55], vision and language integration [56], and of course image segmentation [57].

However, ViTs can be data-hungry and often require large-scale pre-training on big datasets to achieve top performance. This is extra motivation to not train them starting from random weights, instead to apply them in a transfer learning scenario.

For the sake of completeness, let's take a look at a brief summary of how an image

is processed by a ViT:

Patch Embedding: The input image is first divided into small patches, typically of size 16x16 or 32x32. Each patch is then flattened into a vector and linearly transformed (using a fully connected layer) into a sequence of "tokens". This is analogous to how in NLP, an input sentence is divided into a sequence of word tokens.

Position Embedding: In order to retain information about the position of patches in the original image, positional embeddings are added to the patch embeddings. [1]

Transformer Encoder: The sequence of patch embeddings is then passed through a stack of Transformer encoder layers. Each Transformer layer consists of a multi-head self-attention mechanism and a feed-forward neural network, along with layer normalization and residual connections. This allows the model to learn complex relationships between different parts of the image.

Classification Head: Finally, the embedding of the special "[CLS]" token (added to the sequence of patch embeddings) is passed through a linear layer to produce the final classification output.

2.15.2 DINOv2

In the Meta Research group's April 2023 paper[58], they show that self-supervised pretraining methods can produce all purpose visual features - i.e. ones that work across many distributions without pretraining. This paper builds upon the work of a previous paper from the Facebook AI group which highlighted some very interesting properties of Vision Transformers (ViT) that put their capabilities of semantic segmentation beyond those of convolutional neural networks by using self-supervision. They found that self-supervised ViT features contain explicit information about the semantic segmentation of an image, which does not emerge with supervised ViTs,

[1] Interestingly, the position embeddings are learned during training. For an input image divided into N patches, the position embeddings are a NxD matrix, where D is the dimension of the patch embeddings. This matrix is initialized randomly and updated via backpropagation during training. These learned position embeddings are added element-wise to the patch embeddings before they are fed into the transformer encoder blocks.

nor with convnets. They implemented their findings into a simple model called DINO, standing for self-DIstillation with NO labels. The 2023 paper builds on that by scaling up its training, introducing an automatic pretraining pipeline, illustrated in the figure below, giving us the 2nd version of this model: DINOv2, which is what we shall put to the test in this section using audio data.

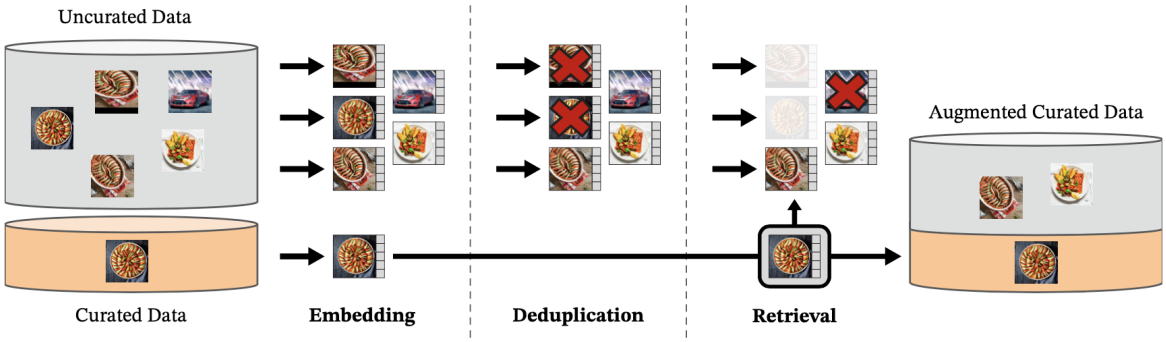

Figure 8: DINOv2 Data Pipeline

2.15.3 SAM

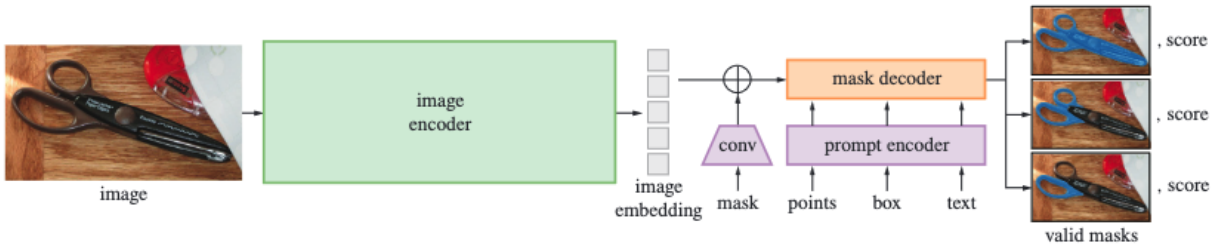

Figure 9: SAM

The Segment Anything Model [59] was introduced by Meta AI Research in 2023 which is an entire project including a dataset, a model, and a new task for image segmentation. The result is the largest segmentation dataset to date, with over 1 billion masks; a promptable segmentation model, which exhibits very high performing zero-shot performance on new image distributions and tasks. It is that characteristic which made it an interesting choice to apply to audio spectrograms.

2.15.4 Architecture

The Image Encoder

The image encoder takes an image as an input and outputs an image embedding that is then used by the mask decoder to make final predictions for the mask. It uses a masked auto-encoder pre-trained Vision Transformer architecture

The Mask Decoder

The mask decoder outputs a mask foreground probability at each image location, using a modified transformer decoder block and a mask prediction head.

The Prompt encoder

The model can accept various prompts, which can be rough masks, points, bounding boxes, or text prompts. The problem of audio source separation is particular, in that separate audio energy overlaps significantly - nearly every source can be represented somewhat in nearly every pixel of the input image. Therefore, at first glance, the idea of using bounding boxes or other positional information may seem to not really apply. However, using this information within the AudioPair dataset, which we will discuss more in-depth later in this paper, is indeed useful in the special case of a mixture which is a foreground and a background sound with distinct density in the spectrogram. In this case, we can use some positional encoding to help with the mask generation.

The useful information would be the ability to use text prompts to separate separate sources. This functionality is not released publicly, and would have to be integrated into the model manually. The paper mentions that it uses a CLIP[60] encoder.

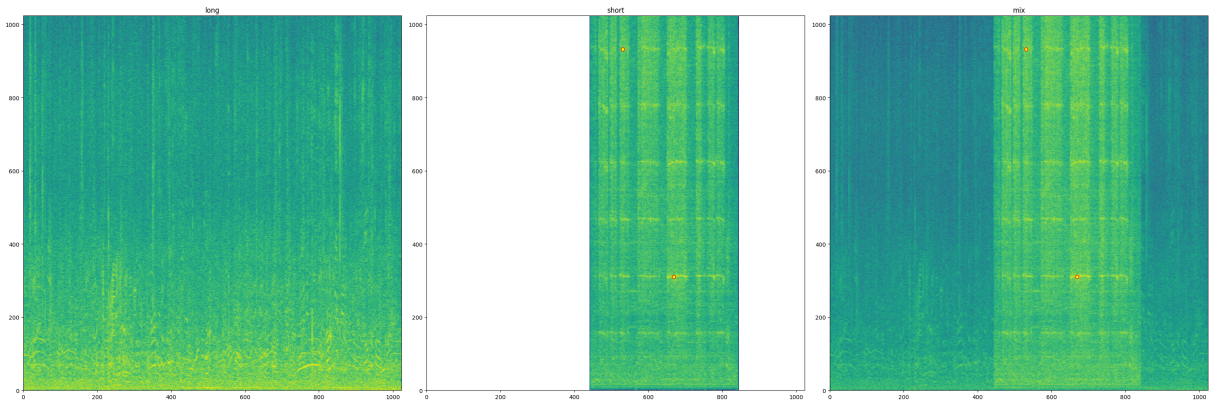

Figure 10: Example input with point prompts

Chapter 3

Network Design

The work of adapting these models to work in a source separation context was to create a software framework in pytorch[61] which acts as an encoder and decoder from- and to- the audio data through the given models.[1]

In order to answer the question whether the pretrained feature extractors are or are not able to extract meaningful information from spectra, we have to be able to make sure that the mask decoders of each are able to translate those features in a way that allows us to quantify those features.

This effectively turns the whole network into one model, in which the segmentation model sits as a pretrained backbone, and the surrounding functions are encoder and decoder parts.

3.1 Architecture

Designing a deep learning model is part art and part science, but the more I understand of it, the more I see that in essence, each layer in a deep learning model can be thought of as a transformation that reshapes the feature space, with the objective of making certain patterns in the data easier to recognize for subsequent layers or systems.

[1] All of the code involved in this project will be available in my personal github at https://github.com/BenjaminOlsen/SegSep.

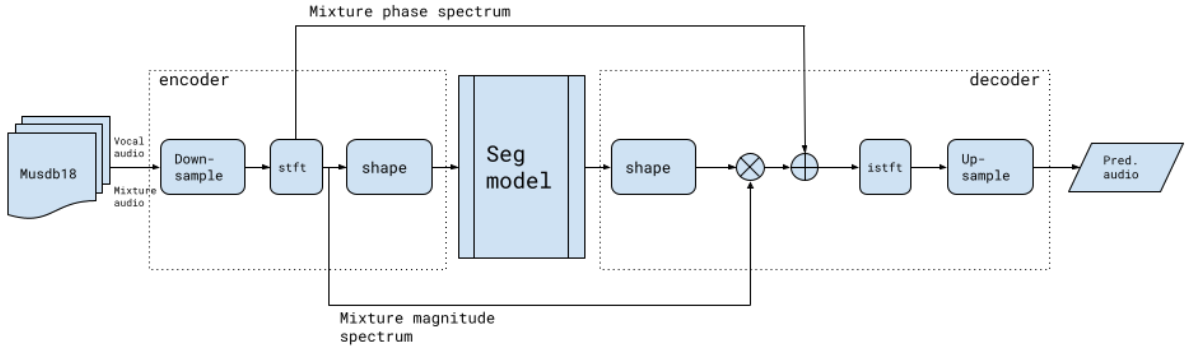

Figure 11: the network

The network in this work consists of a front end encoder which reads audio, divides it into slices, turns each slice into a spectrogram, shapes that spectogram into a tensor for segmentation model, receives the resulting segmentation mask, applies the mask to the input audio's magnitude spectrum, and reconstructs the estimated audio with the original phase from the input audio. This section will describe each part.

3.1.1 Encoder

Downsampling

We downsample to reduce the data requirements, from the 44.1kHz de facto standard sampling rate down to half that at 22.05kHz. While losing information in the high frequencies (above 11.025 kHz), this greatly speeds up the training process, and thanks to the majority of the audio in the FSD50k and MUSDB18 datasets being especially information rich in the lower bands, suffices for our purposes of source separation.

STFT

We turn the downsampled audio into its spectral representation using the Short Time Fourier Transform. The parameters of the STFT are generally very important to be able to clearly represent the important features of the audio in the spectral data, and in this application must be especially carefully constructed in order to minimize distortion of the signal.[2] Each of the models expects input tensors of a certain shape, and attention must be paid to pass the spectral data unchanged through the model.

Explicitly, considering the width and height to be the dimension of the spectrogram:

$$width = \lfloor (audio\_length - win\_length)/hop\_length \rfloor + 1$$

and since we are using one-sided FFTs thanks to audio being real-valued:

$$height = \lfloor n\_fft/2 \rfloor + 1$$

To this end, we specify the width and height of the spectrogram that the STFT should produce, and calculate the fft length to be precisely

$$n\_fft = 2 \cdot (height - 1)$$

We set the window length to be:

$$win\_length = n\_fft$$

The hop length:

$$hop\_length = n_{fft}//8$$

[2]This baseline "identity" contribution of the encoder and decoder without passing through the model is quantified in the Results

and slice the input audio into chunks of size:

$$input\_chunk\_size = (sample\_rate/resample\_rate) \cdot (width - 1) \cdot hop\_length$$

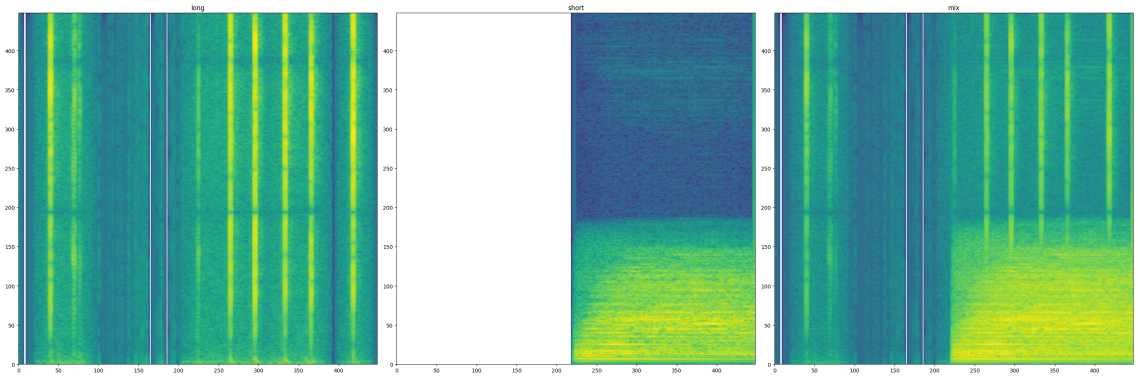

Figure 12: Spectrogram from AudioPair Dataset
(audio 1, audio 2, mixture)

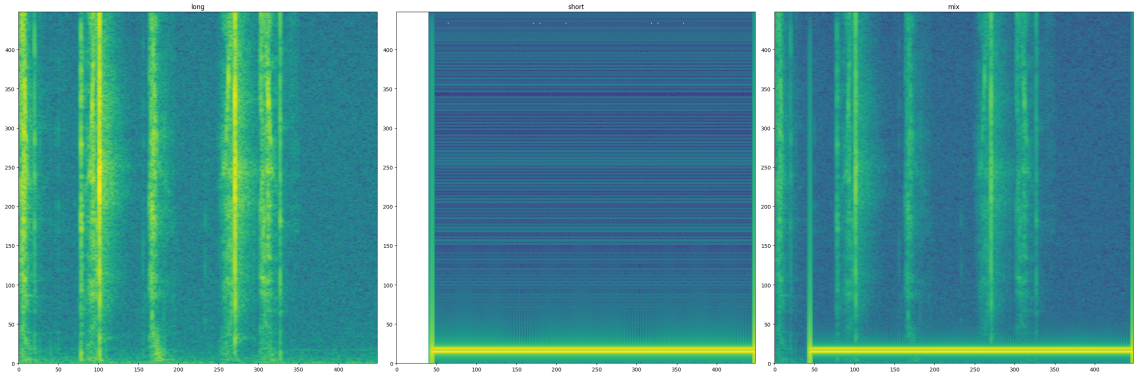

Figure 13: Spectrogram from AudioPair Dataset
(audio 1, audio 2, mixture)

Model Head

The head of the model transform the feature representations (or tokens) output by the vision transformer backbone of the segmentation model, and outputs a segmentation map. This is done differently for each model, as discussed in the state of the art section. Namely, the SAM model comes with a mask decoder pretrained, and the DINOv2 model does not. Therefore, a different training approach must be taken for each one.

For DINOv2, the head must be customized for the intended purpose. As the creators

of the model implied, the DINOv2 is multipurpose visual feature learner, and the goal of the head design is to make a sub-network that can adequately capture the richness of the feature maps produced by the backbone and translate them into segmentation maps.

An initial approach using a linear classifier head was taken, as illustrated in the figure below, where the DINOv2 model's outputs are the Feature Map, which is then reshaped and run through a 1x1 convolutional layer (a.k.a. a point-wise convolution, equivalent to a fully-connected layer), which is then passed through an activation function to generate a predicted mask.

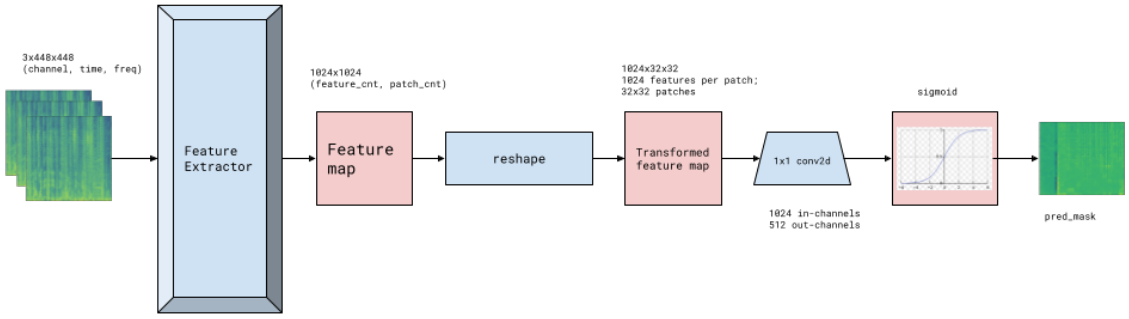

Figure 14: Linear Classifier Head

This approach failed. This is not terribly surprising, as a linear combination of features is the simplest transformation that can be applied, and the nature of visualized sound is quite complex.

Therefore, a more sophisticated head was proposed. This, we call the FeatureTransformer.

FeatureTransformer

There are two sections to the FeatureTransformer: the first is a transformer block that takes the feature map produced by the backbon and passes it through a transformer and convolutional layer. The second part is a stepped convolution-upsampling sequence of operations, which is similar to the expanding path of the U-Net architecture.

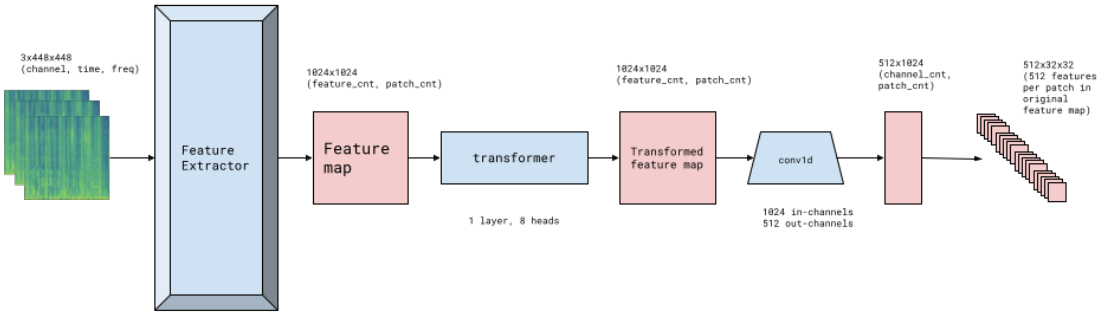

Figure 15: FeatureTransformer pt. 1: transformer block

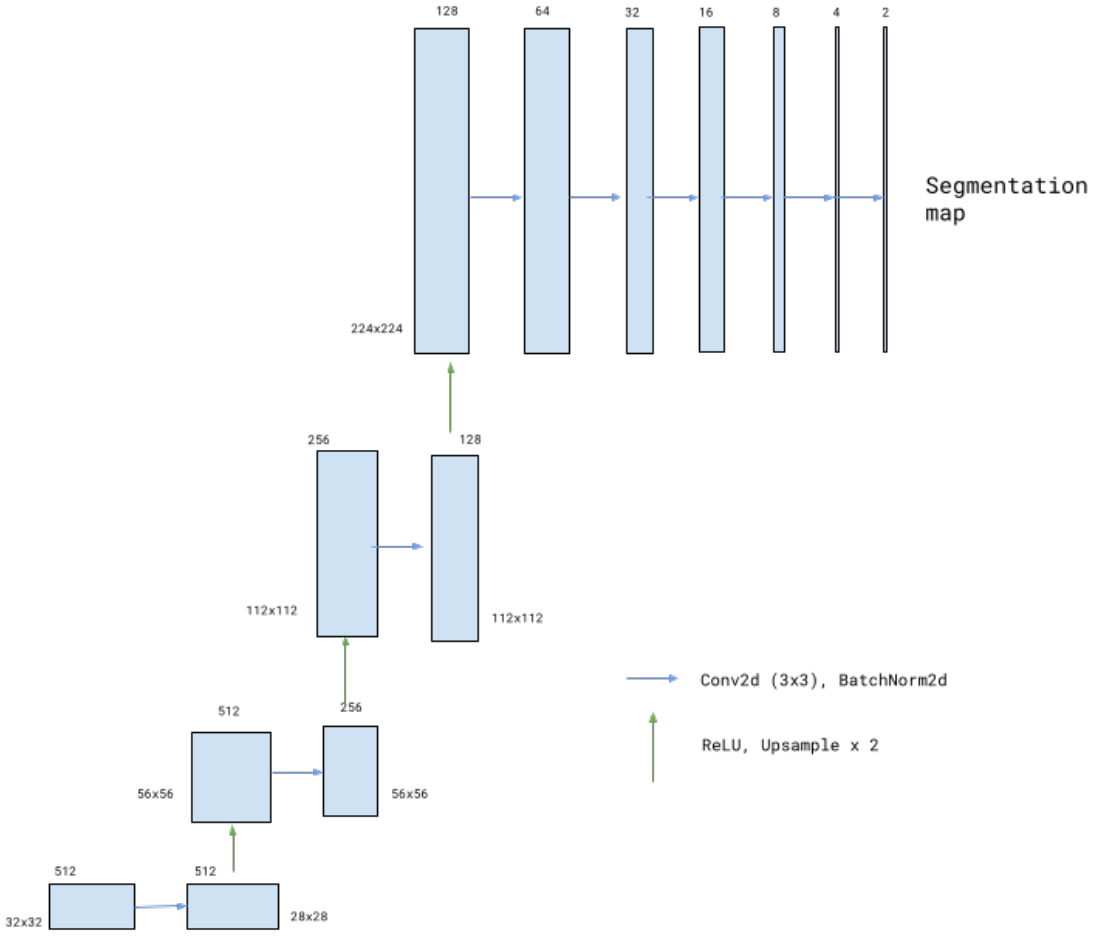

Figure 16: FeatureTransformer pt. 2: upsample-conv

The goal of the transformer block is to provide a learned, non-linear transformation of the input features at each spatial location (or token), which can improve the model's ability to extract useful information from the input data.

In a Vision Transformer (ViT) model, the transformer encoder layers are responsible for modeling the interactions between all pairs of tokens in the input data. This is useful for tasks where the relationship between different parts of the image (the spectrogram, in our case) is important. It's clear that in object detection or semantic segmentation tasks, for example, understanding how different parts of the image relate to each other can be critical for making accurate predictions.

However, in some tasks, it might be beneficial to apply further transformations to the output of the transformer layers. This is where the FeatureTransformer comes into play. It provides an additional, learnable transformation of the features at each token, which can help to enhance the features or bring out certain patterns that might be useful for the task at hand.

In the context of this model, the output of the first part is then passed through a series of convolutional and upsampling layers, which gradually increase the spatial resolution of the feature maps. These operations are well-suited for tasks like semantic segmentation, where the goal is to make a prediction at each pixel (or spectrogram bin). The transformer block helps to provide a good starting point for these upsampling operations, by providing a rich, learned representation of the input data at each token.

In summary, the combination of transformer layers, and convolutional upsampling provides a powerful, multi-scale model architecture that can adapt to a wide variety of tasks.

Decoder

The output of the model and its head is a segmentation map prediction for the given input mixture spectrogram. The result, then is directly applied to the input spectrogram to create a predicted filtered spectrogram for the source. Given a

predicted segmentation mask $\hat{M}$, and input spectrogram S_{in}, the predicted source's magnitude spectrum $|\hat{S}|$ is given by the elementwise product of the mask and the magnitude of the input spectrum.

$$|\hat{S}| = \hat{M} \cdot |S_{in}|$$

This predicted magnitude spectrum is then combined with the phase of the input spectrogram to create an estimated complex spectrum

$$\hat{S} = |\hat{S}| \cdot \cos(\angle S_{in}) + j \cdot |\hat{S}| \cdot \sin(\angle S_{in})$$

which is then passed through an inverse short time fourier transform (ISTFT) using the same parameters as the encoder's STFT to create an real valued estimated source audio $\hat{s}$:

$$\hat{s} = ISTFT(\hat{S})$$

Chapter 4

Datasets

The original aim of this work was to adapt these models to do musical instrument source separation, and the target dataset was MUSDB18, which provides a mixture track along with its separate source stems (vocal, drums, bass, other). However, over the course of the experiments, it was noted that the peformance of the models on this type of dataset was lacking: the quality of the predicted masks were low in the zero-shot scenario, and remained faulty even with fine-tuning.

Instead, we turned our attention to using a Sound Event Recognition (SER) dataset, the FSD50k [62], to which we apply a data augmentation strategy in order to adapt the dataset to audio source separation.

The dataset is split into "dev" and "eval" splits, which we use as the training and evaluation datasets, respectively.

4.1 Data Augmentation

We use a novel data augmentation strategy, inspired by the FUSS dataset [63], which involves building upon the FSD50k dataset to create realistic synthetic mixtures of various sounds for source separation benchmarking. In this work, we introduce the AudioPair dataset which allows for a simplified source separation corpus. The goal was to give a degree of flexibility to the synthetic mixture creation, where instead

of combining sounds selected randomly, we can specify certain conditions on which audio is mixed with which. In this case, we introduce a preprocessing step which collects audio features of each sound in the FSD50k, and then in sampling from the augmented dataset we can specify conditions on that data. In the current work, we set the following conditions:

Those characteristics are: 1) a minimum sample duration, 2) minimum difference in mean spectral centroid between the pair, 3) minimum difference in mean spectral bandwidth between the pair, 4) minimum difference in mean spectral contrast between the pair, 5) minimum difference in mean spectral flatness between the pair.

For each of audio samples in the FSD50k dataset, we collect the mean, maximum, minimum and standard deviation for each of the features.[1], and in the creation of each data point for training, we select two audios that are at least one standard deviation apart in each of the respective features. This has the effect of combining only sufficiently "distinct" audio.

In a general sense, this can be used to customize the "difficulty" of a source separation task. A subtask of this work, would be to find out if there was some sort of threshold across which the segmentation models under consideration would perform better or worse.

The goal was to provide set of training data from which we could move, in evaluation, to a "harder" or "easier" audio dataset with ease. Harder, in this case, would mean that there is more overlap between the foreground and background noise, i.e. a lower minimum difference in the metrics between the pairs; and easier would be the less overlap, or higher minimium difference in the features.

4.2 Audio Features

This section is a brief discussion of each of the aforementioned audio features used in the AudioPair dataset.

[1]See Appendix A for a summary of this data

4.2.1 Spectral Centroid

The spectral centroid is roughly equivalent to the "center of mass" of a slice of spectrum, and corresponds to the perceived "brightness" of a sound, (where a higher centroid typically corresponds to a brighter sound, and a lower centroid is a duller sound), and is a useful quantity in distinguishing the frequency distribution between audio. Its formula C is as follows, given a magnitude spectrum $S(f)$ where f is the frequency.

$$C = \frac{\sum_f f \cdot S(f)}{\sum_f S(f)}$$

4.2.2 Spectral Bandwidth

The Bandwidth is the "width" of a spectrum, which relates to the perceived timbre or texture of a sound, and can have a few different definitions. Here we use the following where given a spectral centroid C, magnitude spectrum $S(f)$ and frequency f, the spectral bandwidth B is:

$$B = \sqrt{\frac{\sum_f S(f) \cdot (f - C)^2}{\sum_f S(f)}}$$

4.2.3 Spectral Contrast

The Spectral Contrast is a measure of the difference between the maximum and minimum values of a magnitude spectrum. A high spectral contrast is more characteristic of harmonic sounds, a lower spectral contrast is more characteristic of noise-like sound, and can be useful for distinguishing between harmonic instruments (like a violin) and percussive instruments (like a drum). [64]

For a given frame of an audio signal, let F_{peak} and F_{valley} represent the magnitudes of the peak and valley frequencies, respectively. The spectral contrast SC for that

frame is:

$$SC = F_{\text{peak}} - F_{\text{valley}}$$

Note: The peaks and valleys are typically determined within specified frequency sub-bands, and is often calculated for multiple sub-bands, resulting in multiple values for each frame. Here, we use the mean value over the entire band, resulting in one value per frame, of which we then take the mean.

4.2.4 Spectral Flatness

Also known as the Wiener Entropy, this is also a measure of how noise-like a waveform is. A value closer to 1 is given by noisier sounds, and closer to 0 more tonal. [65]

The spectral flatness SF is calculated as the ratio of the geometric mean and the arithmetic mean of a spectrum $S(f)$ with N frequency bins:

$$SF = \frac{\sqrt[N]{\prod_{i=1}^{N} S(f_i)}}{\frac{1}{N}\sum_{i=1}^{N} S(f_i)}$$

Chapter 5

Experiment Setup

The experimental phase of this work consists of evaluating the audio source separation performance of the aforementioned models under various conditions.

5.1 Zero-shot

The most straightforward way of evaluating each of the models is in a zero-shot scenario, but this does involve a different approach between the DINOv2 and SAM based models.

SAM Zero-shot

Straight out of the box, so to speak, the SAM comes with a pretrained mask decoder as well as the option to use its prompt encoder's features as well, which can use sparse or dense input - therefore, the possible zero-shot configurations that the SAM based model can take various combinations of these features.

DINO Zero-shot

DINOv2, unlike SAM, does not come with a pretrained mask decoding network, therefore in order to evaluate the zero-shot performance of the feature extracting backbone, the head (described previously) needs to be trained.

5.2 Fine-tuning

The act of fine tuning the respective models involves exposing the components of the model to audio data, specifically training it over the AudioPair dataset described in a previous section. The general approach to training is to train over the entire AudioPair dev dataset, evaluate on a subset of the AudioPair eval dataset. We train using an automatic stopping condition, where we allow for up to 10 epochs of no improvement in the evaluation loss. While the loss is the most important metric, we also track the accuracy as measured by the scale invariant signal-to-noise ratio (SI-SNR) between the estimated audio and the ground-truth audio. You will find the loss / accuracy curves for the training of each model in an appendix.

We use pytorch's Adam[66] optimizer implementation.

SAM fine tuning

As there are multiple components of SAM that can be fine tuned, we have taken various approaches: 1) fine tuning the mask decoder network, 2) fine tuning the image encoder network.

DINOv2 fine tuning

As there is only one component of the provided backbone, fine tuning the DINOv2 model simply involves fine tuning the provided feature extracting backbone.

5.2.1 Loss Functions

Following the work of Enric Gusó[67], we use a log-compressed L2 loss on the magnitude spectra:

$$LOGL2_{freq} = \frac{10}{N\Omega K} \sum_{k} \left| |\hat{Y}_{n,\omega,k}| - |Y_{n,\omega,k}| \right|^2$$

with N the number of frames and. Ω the number of frequency bins in the STFT; and

with $K = 2$ is the number of sources, one foreground sound and one the background sound.

5.3 Evaluation

The evaluation is done over the validation set of the augmented FSD50k dataset, the AudioPairDataset, and the results quantified using the BSS eval package. Given the flexibility of the AudioPairDataset, we can vary the parameters of the generated pairs to see how well the models perform under different audio conditions. To this end, we offer three different sets of parameters for evaluation: easy, medium, and hard. For the easy evaluation, we set the parameters of the AudioPairDataset so that the chosen audios are two standard deviations away from each other. This allows for a greater difference in the character of the sound, and an easier separation task. For the medium condition, we choose the parameters to be one standard deviation apart, and for the hard condition the audios are chosen without setting any minimum distance from one another.

Feature	**std**
Spectral Centroid	1305.68 Hz
Spectral Bandwidth	996.263 Hz
Spectral Contrast	0.0565
Spectral Flatness	0.1054

Table 1: Feature standard deviation summary

The particularity of these conditions is that for each restriction placed on any of the parameters, the effective size of the dataset is changed. The underlying size of the eval set of the FSD50k is 10231 samples, the size of the set of all pairs is $\binom{10231}{2} = 5.23 \times 10^7$. Subject to various minimum length constraints the size of the evaluation datasets is reduced, examples found in the tables below.

These minimimum size constraints are not chosen randomly - they correspond to the length of a single audio chunk for a spectrogram of 448 or 1024 time bins at 22.05kHz: the spectrogram size used for DinoWrapper and SamWrapper evaluation,

Dataset	Sample Count
Easy	2.771×10^6
Medium	1.127×10^7
Hard	4.616×10^7

Table 2: Eval Dataset Size; min length: 4.5s

Dataset	Sample Count
Easy	5.430×10^5
Medium	2.180×10^6
Hard	8.923×10^6

Table 3: Eval Dataset Size; min length: 23.66s

respectively.

In the interest of time, we evaluated each of the models with a random subset of the same size of each respective datasets, limiting each evaluation run to 3×10^3 samples, using the bss_eval_sources function from the mir_eval.separation package [1]

[1] the decision to use this package instead of the museval package was due to the ease of adapting it to this particular case, also since the results don't show a competitive comparison between other source separation tools, the interest of this paper is to show the relative performance of the models trained here.

Chapter 6

Results

The results we'd like to quantify are the standard source separation metrics included in the mir eval BSS package: The Signal-to-Distortion ratio (SDR), Source-to-Interference Ratio (SIR), and Source-to-Artifact Ratio (SAR). An estimate $\hat{s}$ of a source s_{target} is assumed to be comprised of four separate components:

$$\hat{s}_k = s_k + e_{k,interf} + e_{k,noise} + e_{k,artif}$$

where the e terms are the errors attributed to interference from other source, noise, and artifacts, respectively.

$$SDR = 10 \cdot \log_{10} \left(\frac{\sum_k s_k^2}{\sum_k (s_k - \hat{s}_k)^2} \right)$$

SDR evaluates the overall separation quality by comparing the estimated source to the clean source while considering both interference and artifacts.

$$SIR = 10 \cdot \log_{10} \left(\frac{\sum_k s_k^2}{\sum_k e_{k,interf}^2} \right)$$

SIR evaluates the separation between the desired source and the interference or unwanted sources. It quantifies how much the estimated source resembles the clean

source compared to other interfering sources.

$$SAR = 10 \cdot \log_{10} \left(\frac{\sum_k (s_k + e_{k,interf} + e_{k,noise})^2}{\sum_k e_{k,artif}^2} \right)$$

SAR focuses on the quality of the estimated source in terms of artifacts or distortions introduced during separation. It measures how much the estimated source matches the clean source, while accounting for any residual noise.[1]

6.1 Model Summary

Here, we present the set of models which we hold up for evaluation. This is a subset of the models which we trained during the project, but these span the different training strategies used. There is plenty of art[68] in training deep learning models, and so taking a varied approach in training seemed to offer the highest chance for finding the right combination of hyperparameters.[2] In the description of the training strategy in the table below, what is described is the portion of the parameters of the entire model included in the training. Some important details are: DINO1 uses the trained FeatureTransformer from DINO0; SAM3 trains both the mask decoder and image encoder simultaneously, whereas SAM4 first trains the image encoder with a frozen mask decoder in a first run, then freezes the image encoder and trains the mask decoder. The reader may now see that there are many more strategies to be taken, especially considering that SAM's prompt encoder is completely neglected in these runs.

Ideal Model

Even if the model backbone predicts the source's magnitude spectrum mask perfectly, there are still some distortion and artifacts introduced by the series of down-

[1] All of these measurements are presented for the sake of quantitative rigor: they provide *insight* into model performance, but these measurements are not without their issue with respect to source separation quality. Having a better SDR, for example, does not correspond to a "better" separation to the ear, and it could be argued that in the context of audio the *qualitative* separation quality is more important than the quantitative, but of course it depends on the application.

[2] Incidentally, for all of these models we used a learning rate of 1e-5 without slowdown.

Model	Training
IDEAL	Ideal source mask - oracle performance
DINO0	FeatureTransformer head training
DINO1	DINOv2 backbone fine tuning
SAM0	Sam Zero Shot
SAM1	Mask decoder finetuning
SAM2	Image encoder finetuning
SAM3	Mask decoder and image encoder fine tuning
SAM4	Image encoder training followed by mask decoder training

Table 4: Model Summary

sampling, STFT, ISTFT, reconstruction using the phase of the mixture, and upsampling operation. The ideal model, here, is the best possible case wherein the source's magnitude spectrum prediction is perfect. This performance measures the error introduced by everything except the model backbone itself.

6.2 Quantitative Performance Summary

For each of the models, we present the results of the evaluation over the AudioPairDataset's eval set. Each of the values are calculated on a per-track basis.

IDEAL	maximum	minimum	mean	std
Easy, SDR	58.56	-24.58	19.17	14.83
Easy, SIR	286.47	0.00	221.88	40.60
Easy, SAR	58.56	-24.58	19.17	14.83
Medium, SDR	54.23	-18.74	17.21	13.53
Medium, SIR	283.97	0.00	217.80	41.99
Medium, SAR	54.23	-18.74	17.21	13.53
Hard, SDR	54.95	-12.13	15.33	11.02
Hard, SIR	287.01	0.00	213.58	50.88
Hard, SAR	54.95	-12.13	15.33	11.02

Table 5: Summary of Evaluation Metrics for IDEAL

DINO0	maximum	minimum	mean	std
Easy, SDR	51.10	-61.60	2.56	16.88
Easy, SIR	275.40	-91.31	219.39	36.19
Easy, SAR	51.10	-61.60	2.57	16.87
Medium, SDR	47.87	-42.20	-0.81	15.08
Medium, SIR	275.64	0.00	218.23	27.96
Medium, SAR	47.87	-42.20	-0.80	15.08
Hard, SDR	44.79	-52.72	-1.36	12.88
Hard, SIR	276.61	0.00	217.16	30.43
Hard, SAR	44.79	-52.72	-1.35	12.88

Table 6: Summary of Evaluation Metrics for DINO0

DINO1	maximum	minimum	mean	std
Easy, SDR	45.65	-31.55	2.32	17.08
Easy, SIR	273.21	0.00	215.43	34.21
Easy, SAR	45.65	-31.55	2.32	17.07
Medium, SDR	33.67	-36.58	-1.13	15.73
Medium, SIR	271.82	85.40	217.21	25.14
Medium, SAR	33.67	-36.58	-1.13	15.73
Hard, SDR	37.64	-30.10	-1.55	13.28
Hard, SIR	266.77	0.00	213.94	31.56
Hard, SAR	37.64	-30.10	-1.55	13.28

Table 7: Summary of Evaluation Metrics for DINO1

SAM0	maximum	minimum	mean	std
Easy, SDR	20.85	-42.09	-3.84	16.94
Easy, SIR	276.30	-25.41	210.49	53.52
Easy, SAR	20.85	-42.09	-3.84	16.94
Medium, SDR	20.99	-40.24	-4.87	15.14
Medium, SIR	281.61	0.00	206.06	50.81
Medium, SAR	20.99	-40.24	-4.87	15.14
Hard, SDR	18.21	-36.40	-5.39	12.63
Hard, SIR	287.84	0.00	212.03	37.79
Hard, SAR	18.21	-36.40	-5.38	12.63

Table 8: Summary of Evaluation Metrics for SAM0

SAM1	maximum	minimum	mean	std
Easy, SDR	12.59	-46.39	-6.69	14.57
Easy, SIR	274.91	0.00	204.13	59.56
Easy, SAR	12.59	-46.39	-6.69	14.57
Medium, SDR	12.21	-51.69	-7.34	13.08
Medium, SIR	274.99	0.00	205.01	48.72
Medium, SAR	12.21	-51.69	-7.34	13.08
Hard, SDR	10.89	-36.32	-6.99	11.05
Hard, SIR	272.63	0.00	205.66	44.74
Hard, SAR	10.89	-36.32	-6.99	11.05

Table 9: Summary of Evaluation Metrics for SAM1

SAM2	maximum	minimum	mean	std
Easy, SDR	9.52	-43.71	-6.94	13.26
Easy, SIR	275.61	-33.54	208.78	52.66
Easy, SAR	9.52	-43.71	-6.94	13.26
Medium, SDR	10.43	-40.01	-9.65	12.44
Medium, SIR	277.17	0.00	209.95	43.03
Medium, SAR	10.43	-40.01	-9.65	12.44
Hard, SDR	11.74	-40.81	-8.15	10.18
Hard, SIR	281.51	0.00	204.45	50.08
Hard, SAR	11.74	-40.81	-8.15	10.18

Table 10: Summary of Evaluation Metrics for SAM2

SAM3	maximum	minimum	mean	std
Easy, SDR	9.89	-42.63	-7.25	13.15
Easy, SIR	274.16	0.00	207.63	49.33
Easy, SAR	9.89	-42.63	-7.25	13.15
Medium, SDR	9.87	-44.62	-9.08	12.25
Medium, SIR	270.93	-177.44	200.54	51.47
Medium, SAR	9.87	-44.62	-9.09	12.25
Hard, SDR	9.47	-43.05	-8.06	10.07
Hard, SIR	269.98	0.00	197.71	50.56
Hard, SAR	9.47	-43.05	-8.06	10.07

Table 11: Summary of Evaluation Metrics for SAM3

SAM4	maximum	minimum	mean	std
Easy, SDR	18.62	-41.83	-3.75	15.20
Easy, SIR	281.24	0.00	208.61	52.09
Easy, SAR	18.62	-41.83	-3.75	15.20
Medium, SDR	16.01	-35.64	-5.82	13.34
Medium, SIR	281.43	0.00	210.53	40.93
Medium, SAR	16.01	-35.64	-5.83	13.33
Hard, SDR	17.75	-35.31	-5.54	10.82
Hard, SIR	278.62	0.00	204.35	47.78
Hard, SAR	17.75	-35.31	-5.54	10.82

Table 12: Summary of Evaluation Metrics for SAM4

Chapter 7

Discussion

The performance of nearly all of the models in all scenarios is lacking. Surprisingly, we see that DINOv2 out performs SAM in mean SDR.

We can see that the progressive difficulty of the source separation tasks is reflected in the data, where the performance, at least, decreases even further with the difficulty of the dataset.

In a paper from 2020 titled "Black Magic in Deep Learning: How Human Skill Impacts Network Training"[68], which investigates explicitly how human skill is involved in hyperparameter tuning, the researchers come to the conclusion that human skill correlates highly with the final performance of the models. The majority of the work in this has been to find a way to train these models properly, so that they might be able to express their potential for audio source separation. This has not been a trivial task, and by the time of writing of this paper, I can report that there hasn't been a satisfactory settling upon a proper combination of hyperparameters which would allow me to say that the initial goal has been accomplished. Weeks of cumulative training time has been used, without bearing the fruit of a highly performing model. We can see from the results table that while the zero shot performance is lacking, the fine-tuned performance seems even to degrade in parts. I choose to still present the results, even though I cannot stand behind them strongly. I still believe that the potential still lies locked inside them, as I have not been able

to ascertain a definitive reason for their failure to learn the features of audio. This will have to be worked out in the future.

However, I believe that beyond the shortcomings of the models, that this work is not without merit - and in particular, the AudioPairDataset is a worthy contribution to the task of audio source separation. For further work, adding flexibility to this dataset would be highly interesting - for example, adding room impulse response, as is done in the FUSS dataset.

7.1 Final Thoughts

The main work of this thesis is hidden behind the results. I purposefully chose this topic because it went far beyond my area of comfort - at the beginning of this journey I had little to no deep learning experience, but I saw the potential of this project to participate in a hot area of research, to ride a wave of emerging work and to get my hands dirty with the contemporary material of the field. It has been a trial by fire, and although I do not leave it with the satisfaction of having a highly performing model, I have learned more than I expected to at the beginning. And for that, I am truly grateful.

List of Figures

List of Tables

Bibliography

[1] Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding (2019). 1810.04805.

[2] Howard, J. & Ruder, S. Universal language model fine-tuning for text classification (2018). 1801.06146.

[3] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition (2015). 1512.03385.

[4] Vaswani, A. *et al.* Attention is all you need (2017). 1706.03762.

[5] Kaplan, J. *et al.* Scaling laws for neural language models (2020). 2001.08361.

[6] Tan, M. & Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks (2020). 1905.11946.

[7] Ronneberger, O., Fischer, P. & Brox, T. U-net: Convolutional networks for biomedical image segmentation (2015). 1505.04597.

[8] Ali, K. & Hussien, S. Semantic segmentation of aerial images using u net architecture (2022).

[9] Zhang, Z., Wang, Z., Lin, Z. & Qi, H. Image super-resolution by neural texture transfer (2019). 1903.00834.

[10] Huang, H. *et al.* Unet 3+: A full-scale connected unet for medical image segmentation (2020). 2004.08790.

[11] Stöter, F.-R., Uhlich, S., Liutkus, A. & Mitsufuji, Y. Open-unmix - a reference implementation for music source separation. *Journal of Open Source Software* **4**, 1667 (2019). URL `https://doi.org/10.21105/joss.01667`.

[12] Stoller, D., Ewert, S. & Dixon, S. Wave-u-net: A multi-scale neural network for end-to-end audio source separation (2018). `1806.03185`.

[13] Cherry, E. C. Some experiments on the recognition of speech, with one and with two ears. *The Journal of the Acoustical Society of America* (1953).

[14] Bregman, A. S. *Auditory Scene Analysis* (MIT Press, Cambridge, MA, 1990).

[15] Wang, D. & Brown, G. J. (eds.) *Computational Auditory Scene Analysis* (IEEE Press, Piscataway, NJ, 2006).

[16] Auger, F. & Cardoso, F. J. Blind signal separation: Statistical principles (1998). `http://www.cnl.salk.edu/`.

[17] Comon, P. Independent component analysis, a new concept? *Signal Processing* **36**, 287–314 (1994).

[18] Bell, A. J. & Sejnowski, T. J. An information-maximization approach to blind separation and blind deconvolution. *Neural computation* **7**, 1129–1159 (1995).

[19] Hyvärinen, A. Survey on independent component analysis. *Neural Computing Surveys* **2**, 94–128 (1999).

[20] Roweis, S. One microphone source separation. In Leen, T., Dietterich, T. & Tresp, V. (eds.) *Advances in Neural Information Processing Systems*, vol. 13 (MIT Press, 2000). URL `https://proceedings.neurips.cc/paper_files/paper/2000/file/d523773c6b194f37b938d340d5d02232-Paper.pdf`.

[21] Bach, F. R. & Jordan, M. I. Blind one-microphone speech separation: A spectral learning approach. In *Advances in Neural Information Processing Systems* (2005).

[22] Yilmaz, O. & Rickard, S. Blind separation of speech mixtures via time-frequency masking. *IEEE Transactions on Signal Processing* **52**, 1830–1847 (2004).

[23] Reddy, A. M. & Raj, B. Softmask methods for single-channel speaker separation. *IEEE Transactions on Audio, Speech, and Language Processing* **15**, 1766–1776 (2007).

[24] Paatero, P. & Tapper, U. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics* **5**, 111–126 (1994).

[25] Lee, D. D. & Seung, H. S. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems* (2001).

[26] Smaragdis, P. Non-negative matrix factor deconvolution: extraction of multiple sound sources from monophonic inputs. In *International Conference on Independent Component Analysis and Signal Separation* (2004).

[27] Virtanen, T. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE transactions on audio, speech, and language processing* **15**, 1066–1074 (2007).

[28] Schobben, D., Torkkola, K. & Smaragdis, P. Evaluation of blind signal separation methods. In *International Workshop on Independent Component Analysis*, vol. 99 (1999).

[29] Vincent, E., Gribonval, R. & Févotte, C. Performance measurement in blind audio source separation. *IEEE transactions on audio, speech, and language processing* **14**, 1462–1469 (2006).

[30] Vincent, E., Sawada, H., Bofill, P., Makino, S. & Rosca, J. P. First stereo audio source separation evaluation campaign: data, algorithms and results. In *International Conference on Independent Component Analysis and Signal Separation* (2007).

[31] Stöter, F.-R., Liutkus, A. & Ito, N. The 2018 signal separation evaluation campaign (2018). 1804.06267.

[32] Uhlich, S., Giron, F. & Mitsufuji, Y. Deep neural network based instrument extraction from music. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2015).

[33] Venkataramani, S. & Smaragdis, P. End-to-end source separation with adaptive front-ends. *CoRR* **abs/1705.02514** (2017).

[34] Le Roux, J., Wichern, G., Watanabe, S., Sarroff, A. & Hershey, J. R. Phasebook and friends: Leveraging discrete representations for source separation. *IEEE Journal of Selected Topics in Signal Processing* (2019).

[35] Kameoka, H., Ono, N., Kashino, K. & Sagayama, S. Complex nmf: A new sparse representation for acoustic signals. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 3437–3440 (IEEE, 2009).

[36] Dubey, M., Kenyon, G., Carlson, N. & Thresher, A. Does phase matter for monaural source separation? *arXiv preprint arXiv:1711.00913* (2017).

[37] Chandna, P., Miron, M., Janer, J. & Gómez, E. Monoaural audio source separation using deep convolutional neural networks. In *Lecture Notes in Computer Science*, vol. 10169, 258–266 (Springer, 2017).

[38] Grais, E. M. & Plumbley, M. D. Single-channel audio source separation using convolutional denoising autoencoders. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 1265–1269 (IEEE, 2017).

[39] Rethage, D., Pons, J. & Serra, X. A wavenet for speech denoising. *arXiv preprint arXiv:1706.07162* (2017).

[40] Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[41] Jansson, A. *et al.* Singing voice separation with deep u-net convolutional networks (2017).

[42] Lluís, F., Pons, J. & Serra, X. End-to-end music source separation: Is it possible in the waveform domain? In *Proceedings of the Annual Conference of the International Speech Communication Association* (2018).

[43] Radford, A., Narasimhan, K., Salimans, T. & Sutskever, I. Improving language understanding by generative pretraining. *OpenAI* (2018).

[44] Sweeney, L. Discrimination in online ad delivery. *ACM Queue* **11**, 10–10 (2013).

[45] Qian, E., Sagawa, S., Jurafsky, D. & Wu, J. Gender bias in contextualized word embeddings. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2021).

[46] Narayanan, A. Fairness and abstraction in sociotechnical systems. In *Conference on Fairness, Accountability, and Transparency (FAccT)* (2020).

[47] Molnar, C. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable* (Leanpub, 2020).

[48] Halevy, A., Norvig, P. & Pereira, F. The unreasonable effectiveness of data. *IEEE Intelligent Systems* **24**, 8–12 (2009).

[49] De Bièvre, M. & Groshek, J. Collaborative ai: Augmented intelligence for people and business. *European Journal of Marketing* (2021).

[50] Dosovitskiy, A. *et al.* An image is worth 16x16 words: Transformers for image recognition at scale (2021). `2010.11929`.

[51] Carion, N. *et al.* End-to-end object detection with transformers. *European Conference on Computer Vision (ECCV)* (2020).

[52] Durkan, C., Bekasov, A., Murray, I. & Papamakarios, G. Generative adversarial transformers. *arXiv preprint arXiv:2103.01209* (2021).

[53] Bertasius, G., Shi, H. & Torresani, L. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095* (2021).

[54] Berman, M., Rannen Triki, A. & Blaschko, M. B. Fine-grained visual recognition in mobile augmented reality for technical support. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (2020).

[55] Beyer, L. *et al.* Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370* (2020).

[56] Yatskar, M. Clip: Connecting vision and language with localized narratives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021).

[57] Strudel, R., Carion, N. & Timofte, R. Segmenter: Transformer for semantic segmentation. *arXiv preprint arXiv:2105.05633* (2021).

[58] Oquab, M. *et al.* Dinov2: Learning robust visual features without supervision (2023). 2304.07193.

[59] Kirillov, A. *et al.* Segment anything. *arXiv:2304.02643* (2023).

[60] Radford, A. *et al.* Learning transferable visual models from natural language supervision (2021). 2103.00020.

[61] Paszke, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, 8024–8035 (Curran Associates, Inc., 2019). URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[62] Fonseca, E., Favory, X., Pons, J., Font, F. & Serra, X. Fsd50k: An open dataset of human-labeled sound events (2022). 2010.00475.

[63] Wisdom, S. *et al.* What's all the fuss about free universal sound separation data? (2020). 2011.00803.

[64] Jiang, D.-N., Lu, L., Zhang, H.-J., Tao, J.-H. & Cai, L.-H. Music type classification by spectral contrast feature. In *Proceedings of the 2002 IEEE International Conference on Multimedia and Expo (ICME'02)*, vol. 1, 113–116 (IEEE, 2002).

[65] Peeters, G. A large set of audio features for sound description (similarity and classification) in the cuidado project. Project Report, CUIDADO I.S.T. Project (2004).

[66] Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization (2017). 1412.6980.

[67] Gusó, E., Pons, J., Pascual, S. & Serrà, J. On loss functions and evaluation metrics for music source separation (2022). 2202.07968.

[68] Anand, K., Wang, Z., Loog, M. & van Gemert, J. Black magic in deep learning: How human skill impacts network training (2020). 2008.05981.

Appendix A

Statistics for Audio Features

The statistics presented in these tables are for the entire FSD50k datasets, divided into their dev and eval subsets. The way to read the table is that for each frame of each track, each of the features are calculated - therefore, each statistical measure also has a max, min, mean, and standard deviation. The value in at (row, column) is to be read as "the (column) (row) is (value)", e.g. "the maximum standard deviation of the spectral centroid observed across any track of the FSD50k dev dataset is 8458.93"

A.1 FSD50k dev dataset

Centroid	maximum	minimum	mean	std
max	21296.85	0.00	7907.70	3341.01
min	13427.69	0.00	1224.80	1369.01
mean	16767.41	0.00	3425.43	2171.99
std	8458.93	0.00	1305.68	954.93

Table 13: Spectral Centroid stats - dev

Flatness	maximum	minimum	mean	std
max	1.0002	0.0000	0.5934	0.2246
min	0.8052	0.0000	0.0543	0.0839
mean	0.8448	0.0000	0.2135	0.1616
std	0.4134	0.0000	0.1054	0.0685

Table 14: Spectral Flatness stats - dev

Contrast	maximum	minimum	mean	std
max	-0.1588	-1.0000	-0.6476	0.0980
min	-0.6500	-1.0000	-0.9196	0.0484
mean	-0.3864	-1.0000	-0.7887	0.0783
std	0.2696	0.0000	0.0565	0.0327

Table 15: Spectral Contrast stats - dev

Bandwidth	maximum	minimum	mean	std
max	11010.52	0.0000	6201.66	1344.20
min	7297.88	0.0000	1607.50	1153.62
mean	8368.24	0.0000	3493.98	1353.66
std	3566.96	0.0000	996.26	596.22

Table 16: Spectral Bandwidth stats - dev

A.2 FSD50k eval dataset

Centroid	maximum	minimum	mean	std
max	19957.71	300.82	8258.06	3359.88
min	14511.17	0.0000	1123.26	1356.29
mean	16222.08	31.86	3606.87	2206.28
std	7540.10	10.91	1315.95	937.49

Table 17: Spectral Centroid stats - eval

Flatness	maximum	minimum	mean	std
max	1.0001	0.0049	0.6101	0.2251
min	0.7414	0.0000	0.0528	0.0837
mean	0.8182	0.0003	0.2281	0.1662
std	0.4216	0.0003	0.1008	0.0654

Table 18: Spectral Flatness stats - eval

Contrast	maximum	minimum	mean	std
max	-0.2544	-0.9982	-0.6620	0.0956
min	-0.5985	-1.0000	-0.9273	0.0509
mean	-0.4071	-1.0000	-0.8039	0.0705
std	0.3295	0.0000	0.0506	0.0306

Table 19: Spectral Contrast stats - eval

Bandwidth	maximum	minimum	mean	std
max	10647.06	757.15	6244.70	1301.19
min	6831.86	0.0000	1550.14	1214.28
mean	8154.49	44.74	3629.69	1338.46
std	3362.62	31.93	943.64	575.33

Table 20: Spectral Bandwidth stats - eval

Appendix B

Loss / Accuracy curves

These figures show the loss and accuracy progressions per training epoch of each of the models presented in this paper.

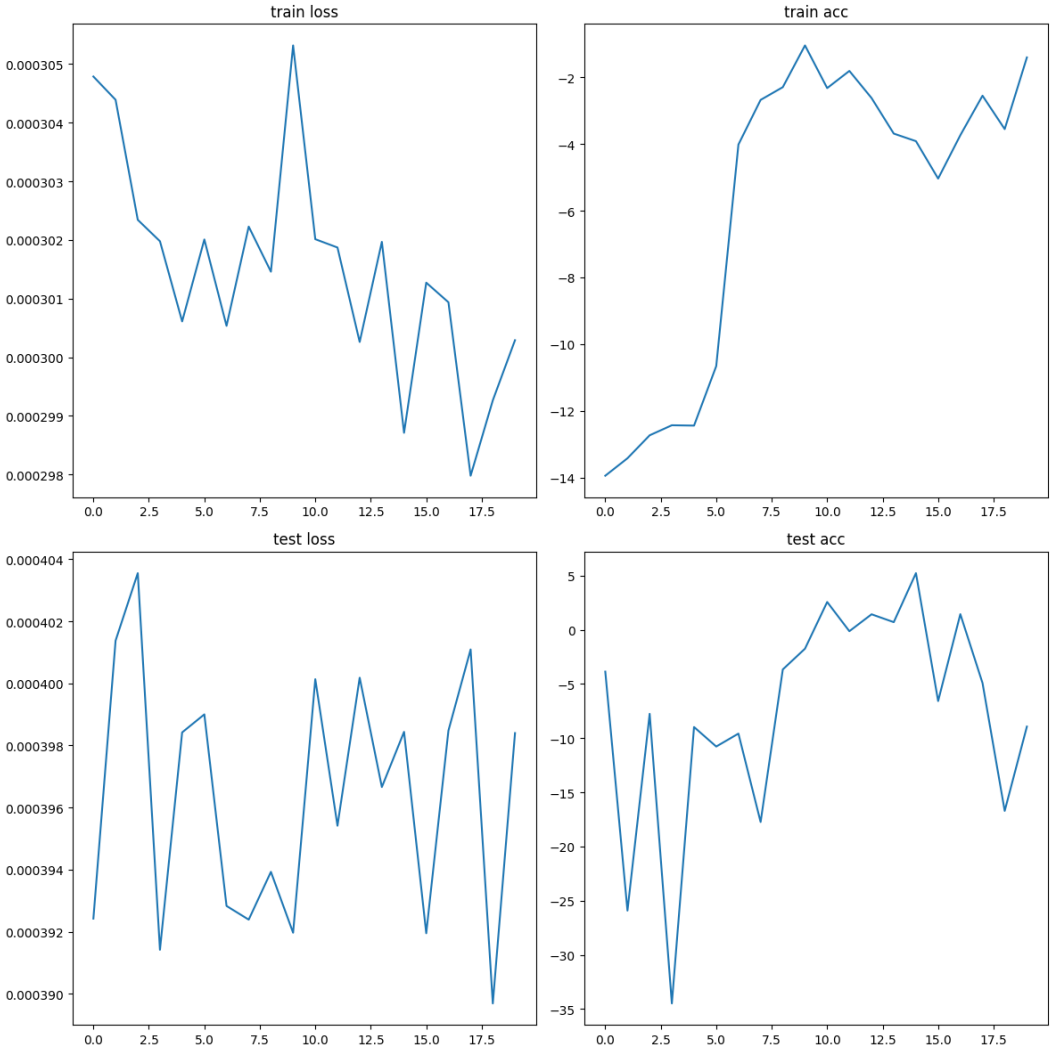

Figure 17: DINO0 loss/accuracy

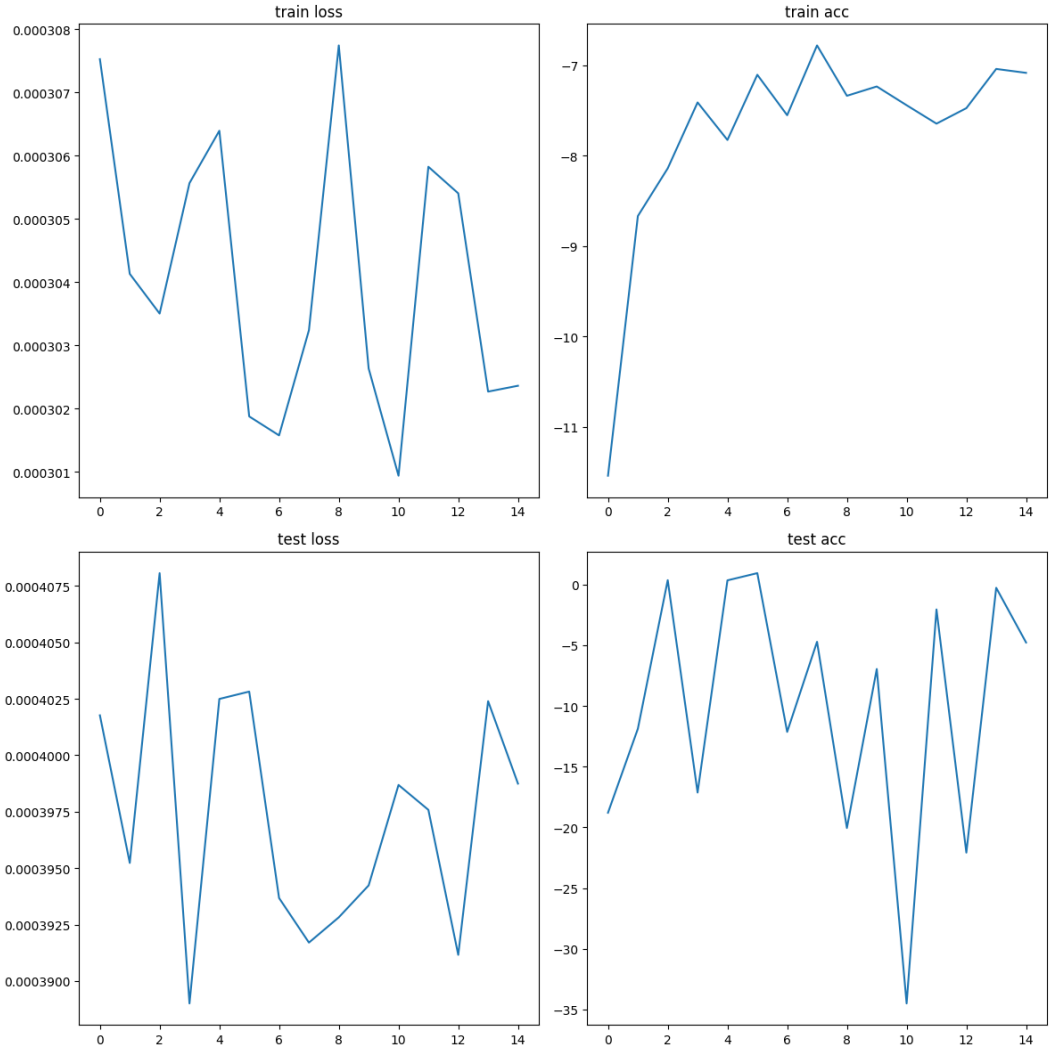

Figure 18: DINO1 loss/accuracy

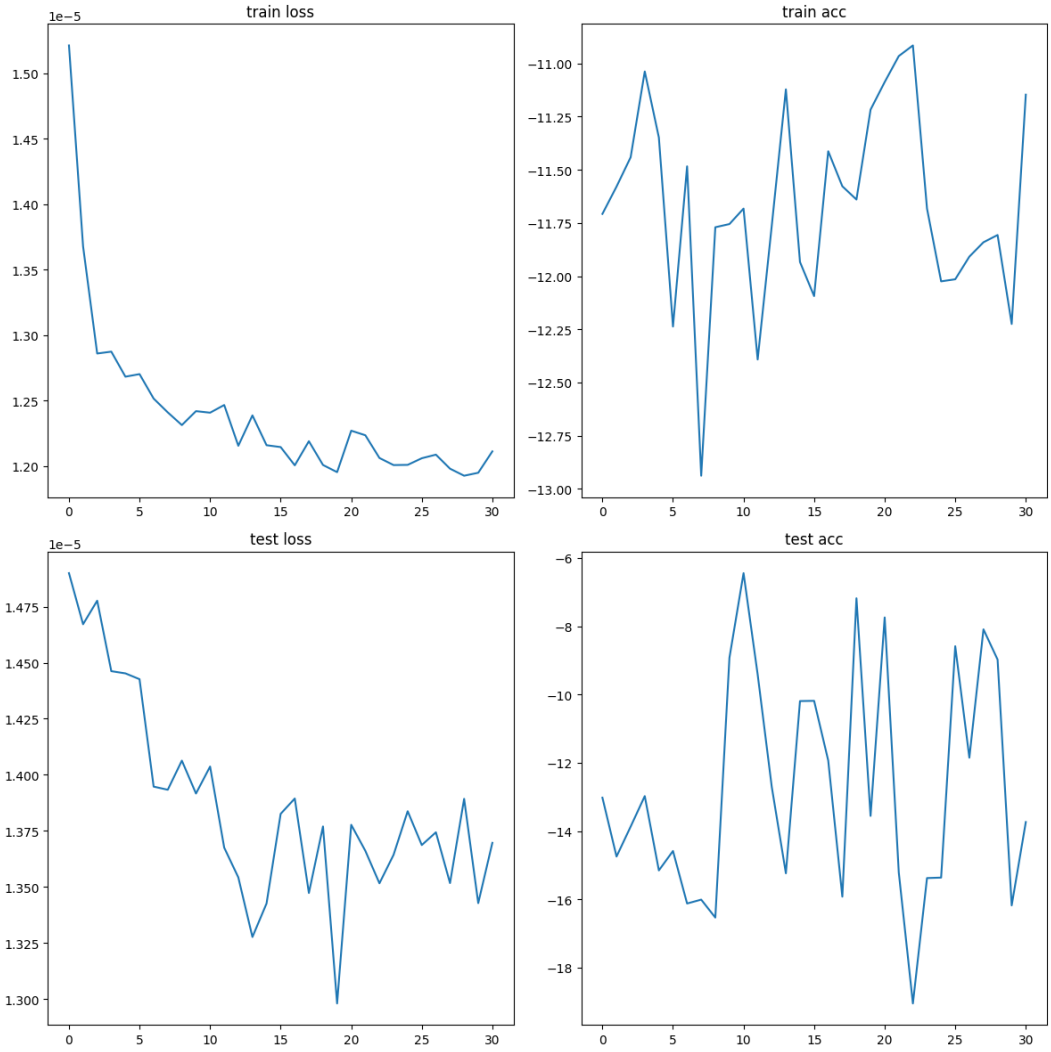

Figure 19: SAM1 loss/accuracy

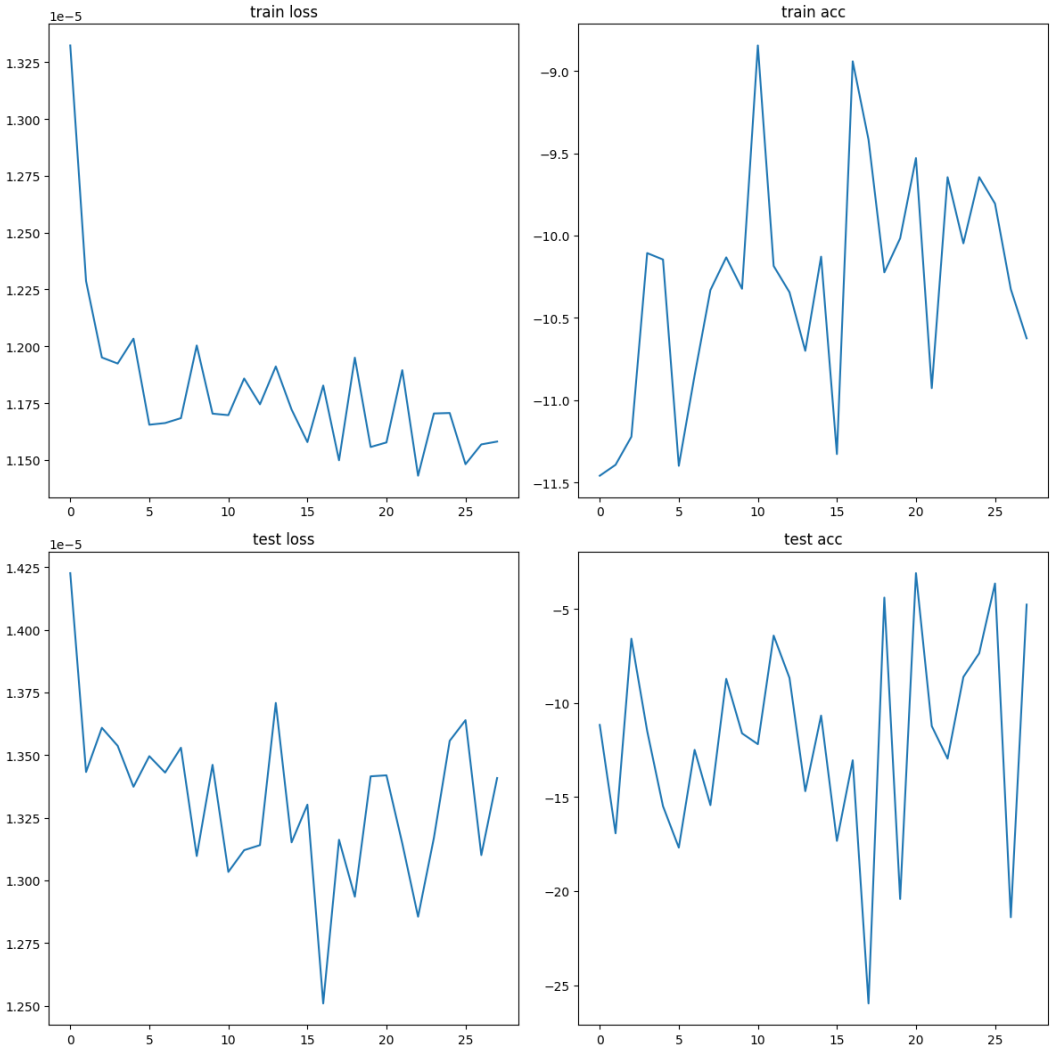

Figure 20: SAM2 loss/accuracy

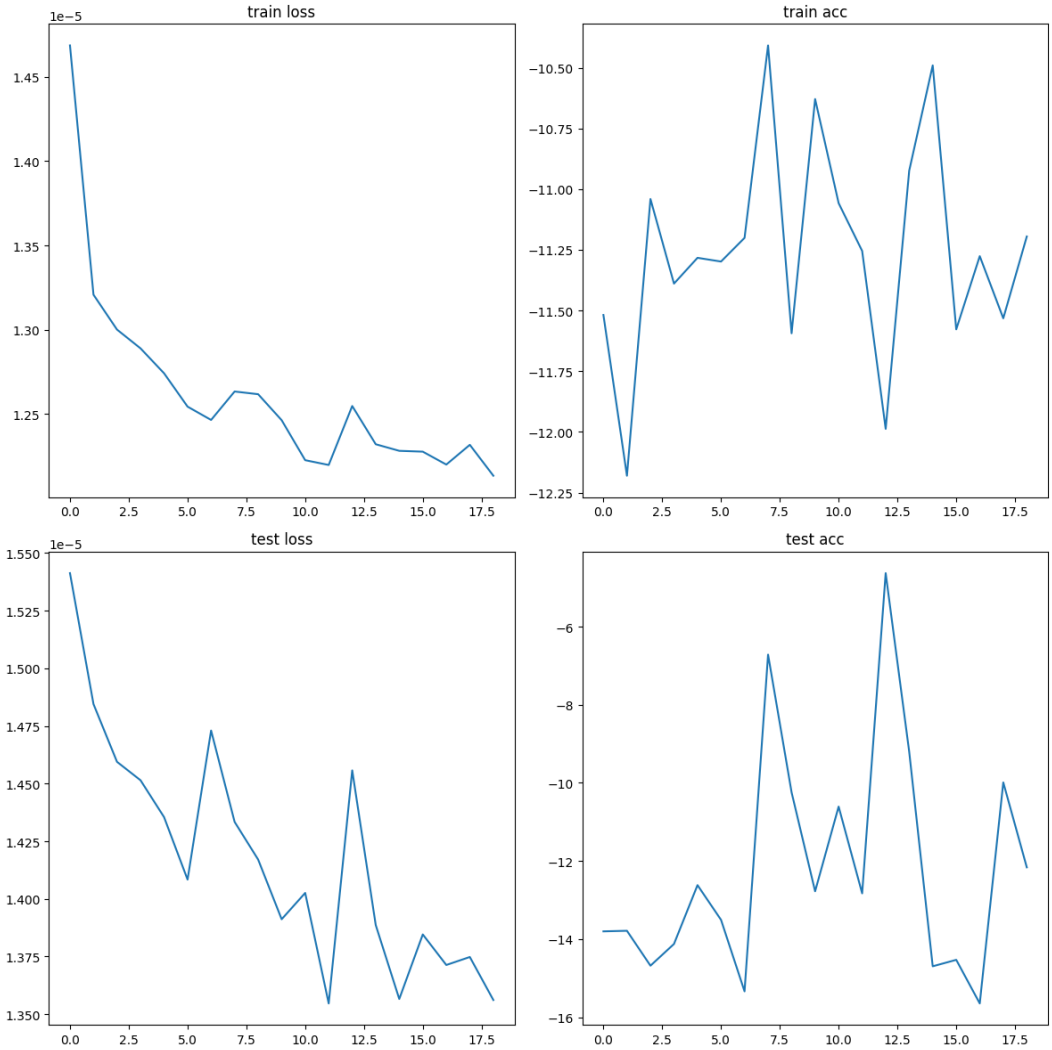

Figure 21: SAM3 loss/accuracy

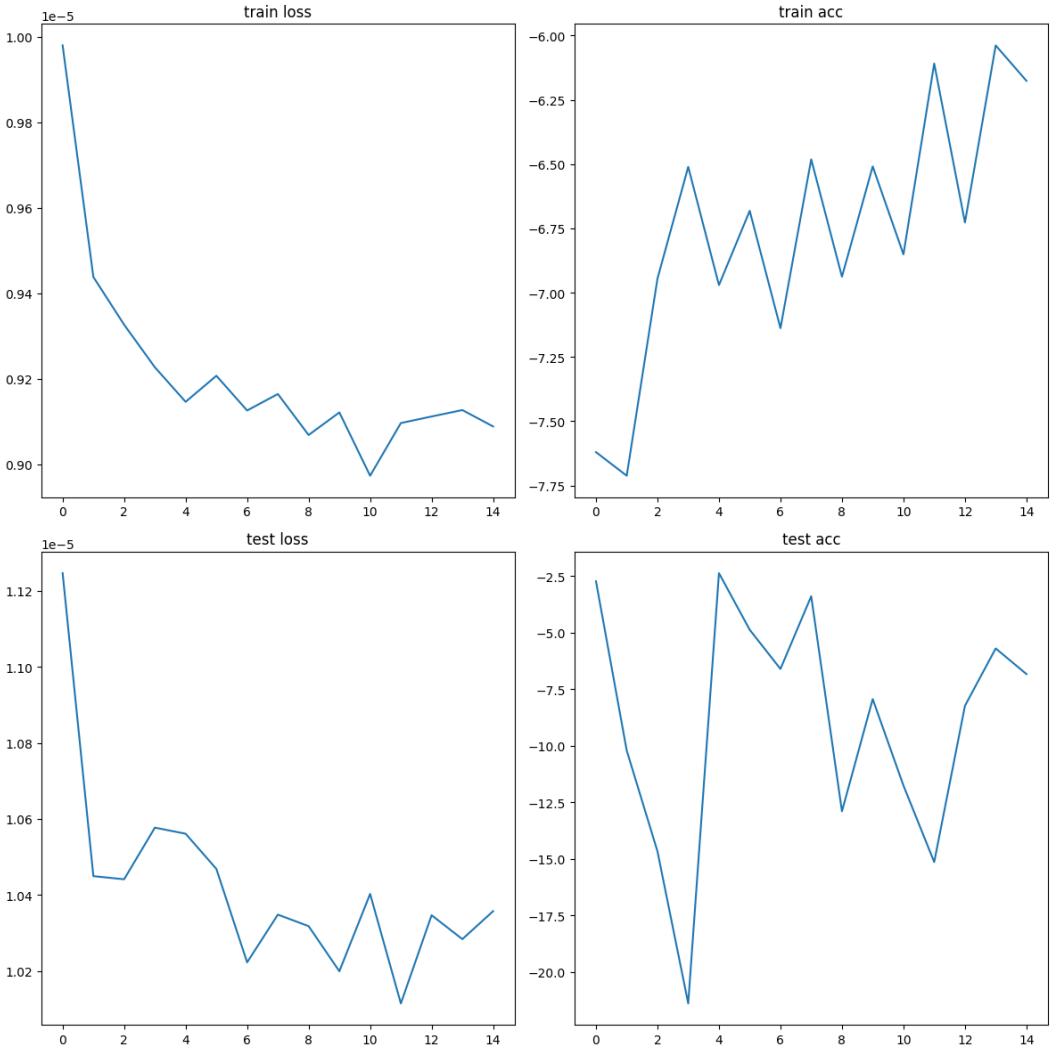

Figure 22: SAM4 loss/accuracy