MUSIC-STAR: A STYLE TRANSLATION SYSTEM FOR AUDIO-BASED REARRANGEMENT

MAHSHID ALINOORI

A THESIS SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE AND ENGINEERING
YORK UNIVERSITY
TORONTO, ONTARIO

DECEMBER 2021

Abstract

Music style translation has recently gained attention among music processing studies. It aims to generate variations of existing music pieces by altering the style-variant characteristics of the original music piece, while content such as the melody remains unchanged. These alterations could involve timbre translation, reharmonization, or music rearrangement.

In this thesis, we plan to address music rearrangement, focusing on instrumentation, by processing waveforms of two-instrument pieces. Previous studies have achieved promising results utilizing time-frequency and symbolic music representations. Music translation on raw audio has also been investigated using single-instrument pieces. Although processing raw audio is more challenging, it embodies more detailed information about the performance, timbre, and dynamics of a music piece. To this end, we introduce Music-STAR, the first audio-based model that can transform the instruments of a multi-track piece into another set of instruments, resulting in a rearranged piece.

Table of Contents

List of Tables

List of Figures

1 Introduction

Artificial intelligence and music, both highly intertwined with our lives, look far apart from each other at first glance. However, their intersection offers considerable potential. One cannot entertain the idea of replacing the extraordinary composers and musicians of human history with machines, but can aim to provide a more convenient way to explore musical creativity, have a richer exposure to musical content, and adopt technology-based approaches for understanding music. For instance, we can see AI-based applications such as AIVA [1], Brain.fm [2], and Amper Music [3], that compose and generate original music pieces, songs, and soundtracks.

Music-related studies have been a topic of interest to musicians and researchers for a long time. Analyzing the content of audio signals goes back to 20th century [68]. For instance, the first attempt on automatic music transcription [74] was made in 1977 where filters were used to detect the note frequencies. [29] However, the

[1] https://www.aiva.ai/

[2] https://www.brain.fm/

[3] https://www.ampermusic.com/

advent of AI-based methods enabled us to go after more complex problems engaging concepts at a higher level than signals and physical features. Today, these problems extend from classification tasks, such as genre classification [101], to novel music and lyrics generation [6, 24]. The recent advancements of deep neural networks in image processing, as well as music information retrieval have created unique opportunities for new applications.

One of the newly emerged subjects is music style translation, defined as manipulating an existing music piece's style-related components, such as instrumentation or performance details, to create variations of the original piece while preserving the content. We have encountered this when listening to a music piece by an artist who has covered another artist's work or has rearranged a masterpiece. Automated music translation allows us to practice musical expressiveness utilizing existing musical content.

Music style translation can be defined in several ways based on what is referred to as style. Some studies address timbre translation [48, 54] where the instruments' timbre is the focus of attention. Timbre, also known as tone color, is the sound quality that helps us distinguish between different instruments, and transforming one to another corresponds to changing the performing instrument.

Other studies address music rearrangement [9, 17, 50]. In music, an arrangement is an adapted version of a previously composed piece. It can be obtained by altering

the original piece in various ways, including and not limited to reharmonization, i.e., altering the chordal accompaniment of an existing melody, instrumentation, i.e., assigning musical instruments to perform different parts of the piece, and altering the piece's structure in terms of the melodic phrases.

In this thesis, the term rearrangement refers to selecting musical instruments different from those in the original performance and automatically deriving an audio performance using the new set of instruments. Based on this definition, there is a correlation between rearrangement and manipulating multiple timbres in music pieces. Therefore, we consider our work as polyphonic timbre translation.

The existing AI music composers, such as AIVA, can help anyone who is not necessarily a musician generate novel pieces based on their preferences. However, if performed at high quality, automatic music rearrangement can assist musicians and composers in experimenting with their pieces instrumentation. It can offer the opportunity to explore new ideas for creating cover songs and help vocalists generate their preferred version of the backing tracks for their songs. In this thesis, we try to take a step toward this goal.

Like many other studies in the audio and music domain that have been inspired by advancements in the image domain, music translation was formed based on image-to-image translation that refers to transforming images from a source style to a target style while keeping the image content unchanged. Image colorization,

season transfer, and image to painting conversion are some of the tasks covered in this area. In recent years, researchers have been practicing the same idea in audio and music to perform voice conversion or music translation.

From a technical point of view, style translation tasks can be portrayed as generation problems with some pre-defined constraints, such as keeping the content unaltered. That is why the effectiveness of generative models, i.e., different autoencoders such as variational autoencoders [5, 9], VQ-VAE [15], and GAN-based [48, 54, 70] architectures, have been investigated in the field.

Some music translation models [48, 54, 70] have treated audio as an image using a three-dimensional representation of audio, known as a spectrogram. Spectrograms are obtained by transforming an audio signal from the time domain into the frequency domain using short-time Fourier transform (STFT). However, this approach poses some challenges and limitations, one of which is transforming the frequency representation back to the time domain and constructing the output audio. The existing methods for this task, including inverse STFT and the Griffin-Lim algorithm, are not efficient enough and can affect the output quality drastically.

This problem has been encountered in many other music/audio-related studies and has motivated the researchers to investigate other means. Some studies [9, 17, 71], have tried to model music using symbolic representation, such as Musical Instrument Device Interface (MIDI). MIDI was introduced as a standard protocol to

allow digital musical instruments, computers, and other tools to communicate with each other. As a file format, MIDI stores musical events and transmits messages to determine the system's behavior and how things should be played. Symbolic representation has been widely used in music generation and translation studies where the models output MIDI that can be played after being converted to audio. Such conversion is made possible using tools such as digital audio workstations (DAW) and virtual software instruments.

Despite being easy to work with, the audio generated from MIDI usually sounds so unnatural that one can distinguish between a virtual performance and a real one. Besides, there are many details in actual performances that MIDI data do not necessarily represent. Consequently, these limitations have created the demand for working with raw audio.

As exciting as it sounds to deal with raw audio, it brings about complications due to its high resolution. In order to clarify the extent of such a challenge, we can compare the size of an image with the size of a digital audio file while training deep neural networks: If we use 720x720 px images to train a network, each image is presented by roughly 520,000 pixels. On the other hand, a two-minute song at CD quality (44.1 kHz and 16 bps) includes over ten million samples and makes it extremely difficult to capture high-level features and long-range dependencies of the audio.

Despite this, speech processing studies [47, 79, 89] developed deep architectures to deal with raw audio. However, the advent of a generative model for raw audio synthesis called WaveNet was a breakthrough in the area. Inspired by autoregressive image generation models [78], WaveNet succeeded at increasing the receptive field and learning long-range dependencies. Conditional audio synthesis was also a novelty introduced by WaveNet, which allows a higher quality in speech synthesis when the network is conditioned on text or linguistic features.

The WaveNet vocoder was then employed in a WaveNet autoencoder [31] to perform music note synthesis. Utilizing the autoencoder architecture, Mor et al. [75] introduced the only audio-based music translation network currently published. Their experiments demonstrate that their model can successfully translate one domain of music, i.e., one instrument track, to another, by changing the timbre.

Although their model can effectively translate an arbitrary instrument track into six different target domains, this question remains unanswered: **what if more than one instrument is in the source piece?**

In this thesis, we explore the idea of multi-instrument translation through which we aim to perform audio-based rearrangement. Instead of single-instrument translation, we quest after a model to translate multiple instruments and provide a new arrangement of the music pieces by redoing the instrumentation.

To this end, we have put forward multiple approaches. The first solution that

suggests itself is to apply the single-instrument translation model to the separate tracks of the piece, transform each of them into a target domain, and put them together to generate the final arrangement. Nonetheless, this approach is only practical when the separate audio tracks of the input mixture are available, which is a rare case as we nearly always have access only to the final mixed and mastered audio.

Consequently, another solution may arise: separating the input audio tracks using music source separation models and applying the single-instrument translation to the isolated track. Although it seems to be a feasible solution, it still adds an extra step which might hurt the output's quality since the source separation models are not flawless.

In this thesis, we propose Music-STAR, a model designed explicitly for multi-instrument music translation that facilitates audio-based rearrangement. Music-STAR is based on the WaveNet autoencoder and offers two training approaches, unsupervised and supervised. In the unsupervised setting, we do not engage the target performance of the source pieces in the training phase, and the pre-trained single-instrument translation models assist us in the task. Conversely, the supervised setting depends on the availability of paired data in the training set.

Since none of the existing audio datasets satisfy the criteria we seek for training our model, we create StarNet, a dataset that includes music pieces comprising two

instrument tracks and their stems performed by two combinations of instruments, i.e., strings-piano and clarinet-vibraphone.

We evaluate the performance of Music-STAR by comparing its results to the baselines that we named as potential solutions to our research questions, i.e., applying single-instrument translation to separate tracks or forming a pipeline of source separation and single-instrument translation to account for the cases where only mixture inputs are available. We assess the resulting outputs based on content preservation, style fit, and audio quality through qualitative and quantitative evaluation methods, exhibiting Music-STAR's success in performing multi-instrument music translation.

1.1 Research Contribution

In our research, we investigate several approaches to address multi-instrument music translation that enables us to perform automated music rearrangement. Our research contributions are:

- The introduction of the StarNet dataset containing 9 hours of two-instrument music pieces performed by two sets of instruments, strings-piano, and clarinet-vibraphone.

- The extension of supported instruments in the state-of-the-art audio-based

single-instrument music translation model through finetuning the model on the StarNet dataset.

– The extension of supported instruments in the state-of-the-art audio-based source separation model through retraining the model on the StarNet dataset.

– The introduction of an unsupervised and supervised approach towards multi-instrument translation based on a WaveNet autoencoder architecture.

– The demonstration of our proposed model superiority to the baseline models through two evaluation methods based on three criteria: content preservation, style fit, and audio quality.

1.2 Thesis Outline

This thesis is structured into five chapters. The first chapter provided an overview of our research problem, our path towards the possible solutions, and our work's major contributions. Chapter 2 discusses the related literature and concepts that illuminate different aspects of our research. In Chapter 3, we present the StarNet dataset and our proposed solution to the problem of multi-instrument rearrangement. The evaluation of our model compared to the baseline models is outlined in Chapter 4. We finally conclude our research with the key findings of our research and discuss the potential future work in Chapter 5.

2 Background

In this chapter, we discuss the relevant studies that have paved the way for us to conduct this research. We first look into the progression of deep neural network architectures through time and some of their applications in the pioneering domains, such as image processing; then, we discuss their applications in the audio domain focusing on music processing. We also present a summary of the existing research on music style translation and other topics that play a role in the configuration of this research.

2.1 Artificial Neural Networks

Machine learning is the study of enabling computers to learn from data. Although some statistical models such as Markov chains had been discovered, the first official learning studies go back to 1950 when Alan Turing proposed the idea of The Imitation Game, asking the critical question of whether computers can think. Since then, machine learning has been extensively studied and has become one of the

most popular techniques in artificial intelligence for task automation. It has found its way into our lives through social networking, medical diagnosis, online shopping, and many other applications. Numerous models have been proposed so far, some of which have gained a significant amount of popularity - support vector machines [19], and random forests [43], to name a few.

Today, the most advanced machine learning models are based on deep neural networks, the idea of which was discussed even prior to learning algorithms. Inspired by the networks of neurons in the brain, artificial neural networks were first introduced in 1943 when a mathematical model was proposed in [72] on how the brain's neurons communicate and perform logical processes. The idea of a simple neural network like a perceptron was explored in 1957 by Frank Rosenblatt and then was elaborated into multilayer perceptrons (MLP) by Alexey Ivakhnenko in 1965, which was a starting point for deep architectures.

In the following years, the early versions of other network architectures such as convolutional neural networks [33], and recurrent neural networks [45] were discovered. However, neural networks could not prove their true potential due to limited computational power and issues related to training. An important training issue was solved when the backpropagation algorithm that was first introduced in [59] was successfully applied and explained in 1985 by Rumelhart et al. [86]. The combination of backpropagation with gradient-based learning [65] was another

groundbreaking innovation that greatly helped in training the neural networks.

Since then, the considerable number of studies for improving deep architectures and the increase of computational powers caused by the utilization of Graphical Processing Units (GPUs) have largely contributed to making deep learning the state-of-the-art approach in AI world. In the following sections, we will briefly discuss a variety of neural network architectures, namely convolutional neural networks, recurrent neural networks, autoencoders, and generative adversarial networks, along with their applications in music studies relevant to this research.

2.1.1 Convolutional Neural Networks

Similar to neurological connections having inspired artificial neural networks, convolutional neural networks, first seen in [33], specifically resulted from modeling the brain's visual cortex. According to physiological and psychological studies, visual perception occurs in multiple stages, each of which is addressed by a specific region in the brain. The studies also show that the neurons in the different parts of the visual cortex have different responsibilities. Some have smaller receptive fields and process simple visual forms like edges, while others with larger receptive fields perceive higher-level features, such as detecting objects. The same scenario applies to convolutional neural networks. They recognize the patterns in the input signal in a hierarchical manner in which the initial layers capture low-level patterns and

deeper layers learn the higher-level ones.

Besides, the natural signals' properties such as stationarity, locality, and compositionality [8] make the convolutional neural networks efficient in computations by maintaining parameter sharing and sparsity. These properties remove the need for fully connected layers in the networks leading to a limited receptive field and fewer inter-layer connections.

LeNet [66], the first convolutional network trained by the backpropagation technique, was used to recognize handwritten digits. Sub-sampling layers that today we know as pooling layers were first used in LeNet architecture.

AlexNet [61], the next groundbreaking CNN model, managed to achieve state-of-the-art results in classifying the images from the ImageNet dataset, which has become one of the benchmark datasets in image processing research. While adopting more layers in the network's architecture, the authors took advantage of multiple training techniques such as regularization, dropout [95], and Rectified Linear Unit (ReLU) activation.

In the following years, researchers worked on improving CNNs by exploring more complex and deeper architectures. The benchmark datasets grew, and the advancements of GPUs were definitely a significant help. By 2015, multiple versions of VGG [93] and Inception [99] networks were introduced, each of them increasing in the number of layers and becoming more powerful in image classification.

The residual blocks and skip connections that were first adopted in ResNet [42] succeeded in overcoming some training complications, such as vanishing gradient caused by the large number of layers, enabling the networks to go even deeper. Xception [14] was the first network that took advantage of depth-wise separable convolutions to reduce the number of parameters.

CNNs are also capable of performing image segmentation, i.e., detecting the objects in an image and classifying the pixels accordingly. Image segmentation is remarkably helpful in biomedical image diagnosis. U-Net [84] is a successful image segmentation model with a u-like architecture consisting of two paths. In the down-sampling path, the network applies convolutional layers along with pooling layers to increase the receptive field and capture the context of the image. In the upsampling path, transposed convolutions, also known as deconvolution layers, are applied to retrieve the original image resolution and the localization of the objects.

Today, CNNs are successfully adopted in real-life use cases such as face recognition, visual search, medical image analysis, self-driving cars and have been applied to many areas other than computer vision.

2.1.2 Recurrent Neural Networks

Recurrent neural networks (RNNs) emerged to model temporal dependencies observed in sequential data such as speech, text, DNA sequences, and music notes.

Although there are successful models that address temporal data using CNNs, some of the limitations of the convolutional neural networks, like the fixed length of input and output was a reason to start seeking more adaptable models. Hopfield Network [46], the very first recurrent neural network, was developed in 1982. The author introduces the idea of maintaining associative memory so that the network can store some patterns and recall them when necessary.

RNNs are able to handle inputs and outputs of different sizes. While sharing the parameters for all the time steps, the hidden states carry information from one step to the next. The hidden state at every step is obtained by processing the input and is then fed to the next step. Hidden states represent the idea of memory as they pass on information, but they cannot account for long-term memory if the number of layers in the network increases, mainly caused by vanishing (exploding) gradient in which the gradients' values get so close to zero (infinity), while the high number of layers get trained through backpropagation.

Long Short-Term Memory (LSTM) [44] recurrent neural networks were suggested as a solution for this issue. The main idea behind the LSTMs is to only remember the relevant information for as long as it is necessary instead of constantly maintaining a long-term memory that may contain unnecessary information. Every cell in the LSTM decides what information to pass to the next step as the memory, what information to output, and what to forget using three different neural net-

works called gates: the input gate, output gate, and forget gate. LSTMs have been mainly used in speech recognition [38], and sequence-to-sequence learning applied to machine translation [98] both in academic research and commercial products.

In 2014, Gated Recurrent Unit (GRU) [13], a variant of LSTMs was introduced. By merging the cell-state and the hidden state, GRUs reduced the number of gates to two, which results in fewer parameters and thus less complexity. This property makes GRUs faster in training and a better candidate for cases when less training data is available. Similar to LSTMs, GRUs have also been employed in state-of-the-art systems for sequential modeling.

LSTMs and GRUs can only use the information from the past to predict the future. However, there are cases that useful information might be extracted from future data and bidirectional recurrent neural networks [91] can help in such cases. In bidirectional RNNs, there are two recurrent neural networks, one of which passes the information from the past to the future, and the other carries them from the future to the past by processing the inverted sequences of the input.

Bidirectional LSTMs have been successfully employed in speech recognition and text-to-speech systems. One of the exciting applications of BiLSTMs has been seen in PixelRNN [102], designed for modeling the joint distribution of natural images and generating life-like images. According to their autoregressive model, the joint distribution of every pixel can be estimated as the product of the conditional

probabilities of the preceding pixels, and that is how they modeled an image as sequential data. However, RNNs are slow both during training and inference due to their sequential nature; that is why a CNN version of the model was also presented as PixelCNN that was soon promoted to generate images conditionally [78].

As mentioned before, sequence-to-sequence models for neural machine translation are one of the well-known applications of RNNs. In machine translation, the input and the output do not necessarily have the same length. The sequence-to-sequence model [98] was proposed to handle such language modeling problems with an encoder-decoder architecture where both the encoder and the decoder are made of RNN units. In the case of machine translation, the encoder comes up with a context vector of the sentence from the source language, and the decoder predicts the translated sentence in the target language.

The attention mechanism [103] was first introduced to provide an alignment model between the input words and the output words so that the decoder can have access to any valuable data that has been forgotten and is missing in the context vector. Since then, attention has been applied to many deep learning models to direct the network toward the important parts of the input data.

2.1.3 Autoencoders

Autoencoders are one of the types of neural networks that have been popular for their applications in data reduction, feature learning, denoising, and building generative models. Today, they are among the most effective frameworks in deep architectures and are largely engaged in unsupervised learning. They were first introduced in [87] to address the problem of mapping the input to the output through the internal representations learned by the hidden units in the network.

Autoencoders consist of three components: the encoder, the code, and the decoder. The encoder takes the input and provides a meaningful reduced representation, also known as the code, based on which the decoder can reconstruct the input. However, some constraints should be placed either by minimizing the size of the code or adding some noise to the input data to prevent copying the input to the output via an identity function that prevents learning meaningful features. Deep autoencoders are obtained by employing deep networks in implementing the encoder and the decoder and can vary in the architecture resulting in feedforward, convolutional or RNN-based autoencoders.

Autoencoders were traditionally used for dimensionality reduction by limiting the code size. The curse of dimensionality and its complications have consistently forced researchers to adopt dimensionality reduction techniques, and autoencoders

became a serious rival for long-standing methods such as Principal Component Analysis (PCA). Undercomplete autoencoders with lower dimensionality in the code than the input provide the dimensionality reduction in their bottleneck layer, and can handle more complexities because of the presence of neural networks. [2]

Adding noise to the input data and training the autoencoder to generate clean data by removing the corrupted parts have been investigated in Denoising Autoencoders [105]. This technique is either used as a general training technique or an independent application for error correction. [2]

Although autoencoders are well-known representatives of unsupervised learning, they also contribute to semi-supervised settings. The encoder plays a role as a feature extractor, and the obtained representation is used for classification purposes under the assumption that the encoder outputs similar representations for the data samples from the same class. [2] Discovering the correlations between data points based on the encoded representations has made the idea of autoencoder-based recommender systems possible [92, 62].

Variational Autoencoders (VAEs) [60] brought the autoencoder framework into the world of generative models. As a probabilistic generative model, VAEs learn the underlying probabilistic distributions of the features in the input data and represent them in the code produced by the encoder. Then the decoder randomly samples from the learned distributions in the latent representation and generates an unseen

datum. The reparameterization trick is used to make backpropagation possible for random sampling. VAEs applications have been explored in the domains of image, text, and audio generation.

2.1.4 Generative Adversarial Networks

Generative adversarial networks (GANs) [37] are a group of generative models that learn to implicitly capture the underlying distribution of the data and sample new ones from the learned distribution. GANs are composed of two neural networks: the generator and the discriminator. These two networks play a minimax game against each other. Take image synthesis as an example. In this case, the generator tries to generate a realistic image given a random distribution without seeing any images from the training set. On the other hand, the discriminator has access to the real data and tries to distinguish the generated image from the real ones. Using the feedback it receives from the discriminator, the generator should finally learn to output an image similar to the authentic data so that the discriminator cannot tell them apart.

Although the general idea of adversarial networks has caught a great deal of attention, training GANs was not an easy task from the beginning. Throughout the years, researchers have investigated different stabilizing methods and training tricks to mitigate the complications. One of the major concerns is synchronizing

the training of the discriminator and the generator. If the discriminator improves faster than the generator, its loss converges to zero and will not allow the generator to get updated. Another critical issue is mode collapse. This happens when the diversity of generated data samples decreases, and as a result, the discriminator cannot provide constructive feedback resulting in the generator getting stuck in local minima.

The original GAN architecture [37] took advantage of fully connected layers in the generator and the discriminator. Later on, CNNs found their way into GAN architectures, and deep convolutional GAN (DCGAN) [81] overcame the difficulties of training deep convolutional layers by applying tricks like strided and fractionally strided convolutions, batch normalization, ReLU, and Leaky ReLU activations.

The generative recurrent adversarial network (GRAN) [52] was the first GAN architecture with a generator consisting of recurrent feedback loops designed for image synthesis. Bidirectional GANs (BiGAN) [27] introduced the idea of adversarial feature learning by adopting an encoder to map the input to a latent representation and asking the discriminator to judge based on both the generator's and the encoder's outputs.

As of today, hundreds of GAN architectures have been proposed to adapt the adversarial networks to various domains and applications. So far, GANs have learned to create novel artistic pictures by deviating from the style of an input picture as

seen in the CAN [28] model. GANs are applied to a variety of techniques for image editing and manipulation [107, 7] especially for facial editing [1, 25]. GANs have managed to synthesize images based on text descriptions [83]. GANs are used to upscale and enhance the quality of photos and videos, even the ones from a century ago. They are able to generate videos of people speaking only by seeing their photos and are well-known for being at the forefront of deepfake technologies.

2.1.5 Image-to-Image Translation

Image-to-image (I2I) translation is a class of computer vision tasks that aims to translate images across domains. It takes various forms, such as image inpainting, image colorization, and neural style transfer.There are four approaches toward image-to-image translation according to [80]:

1. Supervised I2I: Using paired images of the source and target domains to train the model, i.e., the same image content before and after translation.

2. Unsupervised I2I: Using unpaired images and learning the mapping between two image collections.

3. Semi-supervised I2I: Using the source images along with a limited number of source-target training data.

4. Few-shot I2I: A few examples adapt a pre-trained network to a source-to-

target translation inspired by transfer learning.

Image style transfer [36] was among the earliest models in this area and explored both photograph-to-artwork and photograph-to-photograph translation using CNNs trained on paired data. Their work was later promoted to preserve the coloring of the source image during the transfer [35].

Recently, GAN-based image-to-image translation techniques have received significant attention. Pix2Pix [53] was the first framework using conditional GAN to translate the style of images in a supervised setting with paired images.

Although it is more challenging to address image translation in an unsupervised setting, unpaired translation models have become popular as they do not depend on a dataset that includes the original image and its counterpart after translation.

CycleGAN [108] addressed this problem by adopting cycle consistency in which the network learns to perform both forward and backward translations. In other words, the network should be able to translate the input image into the target style and also retrieve the original image from the translated version. The effectiveness of this model has been proved in object transfiguration, season transfer, and photo enhancement.

An unsupervised translation network called UNIT [69] was built based on the idea that a pair of corresponding images from two different domains can be mapped to shared latent space. By adopting VAEs, GANs, and cycle consistency, the model

learns to enforce one latent representation to generate images in two domains. UNIT can successfully translate satellite images into maps and perform animal species translation.

UNIT introduced a one-to-one or unimodal mapping between domains. Later, MUNIT [49] was proposed to address multimodal image-to-image translation to capture the distribution of all possible outputs. A key factor in their model is that they decompose the latent space into the content space and style space. Assuming that the images from different domains can only share the content space, MUNIT combines the content space of the source image with a random style code from the target style space to generate diverse outputs.

2.2 Deep Learning in Audio and Music Processing

In the previous section, we discussed deep learning advancements and the pioneering models that mainly manifest in the image domain or natural language processing. This section will delve into the research contributions in audio and music processing and sheds light on how preexisting deep architectures have paved the way for this discipline to grow. But first, we will review the fundamental concepts of audio and music representation used in data modeling. Then, we will present the most relevant applications of deep learning to our research, such as audio synthesis, music source separation, and music style translation.

Music-related studies that fall under the fields of music information retrieval and computer music emerged later than audio and speech processing. These studies are multidisciplinary, engaging multiple areas like signal processing, musicology, psychoacoustics, and machine learning. The recent music processing techniques mainly concern automating machines in understanding, analyzing, manipulating, and even generating music using deep learning models.

The extended accessibility of music through music streaming services, mobile, and web applications has made it possible for us to take advantage of these research achievements. We have encountered new songs based on recommender systems built upon music embeddings; we have used the audio fingerprinting techniques embedded in virtual assistants or applications like Shazam to help us identify the songs we hear in a cafe; and we can use chord recognition applications to learn how to play our favorite songs.

Below are some other music processing tasks that researchers have been working on:

- Music classification: Categorizing music recordings based on their genre (pop, rock, jazz, blues, etc.) [101] or their mood (sad, happy, angry, relaxed) [64].
- Instrument recognition: Identifying the musical instruments played in the music recordings. [32, 41]

– Query by humming or singing: Identifying songs based on the hummed or sang melody. [39, 88]

– Automatic music transcription: The task of extracting score parameters from a music recording like the note events, time signature, key signature, and dynamics. [4, 12]

– Music source separation: Extracting individual instruments and vocal tracks from a mixed recording. [97, 22]

– Automatic music generation: Composition of novel melodies, chords, or full songs without human assistance. [6, 24]

Before the rise of machine learning techniques, developing high-level algorithms was challenging and music-related studies mainly depended on algorithmic analysis and algorithmic composition [85]. For instance, traditional music transcription depended on onset detection, and pitch tracking [3]. Automatic music composition was explored through different techniques, such as Markov chains, genetic algorithms, and pattern-processing algorithms. [18] However, today, the possibility of low-level processing is taken for granted, thanks to deep learning models, and state-of-the-art research has taken up the challenge to deal with audio directly. Despite all the limitations that have remained unresolved, music technology is standing at

the point where systems like AIVA and Amper Music can generate songs corresponding to a specific genre or mood.

2.2.1 Music and Audio Representation

Deciding on how to represent the audio is a determining factor in our model design since it specifies the type of input the neural network should process. Raw audio, time-frequency, and symbolic representations are the most commonly used types of representation that will be discussed below.

2.2.1.1 Raw Audio Representation

Sound is generated when the vibrations of a source like human vocal folds or a string of an instrument propagate through a medium, such as air, in the form of an acoustic wave.

Sound is harmonic in nature. If we listen to a sinusoidal wave, we hear a pure tone with only one present frequency. However, most of the sounds we hear are more complex than a pure tone. When a specific pitch is played by an instrument, it contains other frequencies that are integer multiples of the lowest frequency. The lowest frequency is known as the fundamental frequency, and its multiples are the harmonics.

Sound embodies various properties that play significant roles in how we perceive

it, some of which are explained below:

1. Pitch: Pitch is a perceptual attribute related to the sound wave frequencies. In music, every pitch is notated by a letter and a number. The letter indicates the pitch class, while the number indicates the octave. For example, A1, A2, ..., A8 all belong to the pitch class A. The pitch A4, known as the concert pitch, has a fundamental frequency of 440 Hz, while A3 has a frequency of 220 Hz (i.e. $\frac{440}{2}$) and A5 has a frequency of 880 Hz (i.e. $440 * 2$). Since all the A notes in different octaves are harmonically related and share a unique mathematical relationship, they are perceived similarly by the human brain. This quality applies to all notes and is the reason that the same notes are repeated in every octave.

2. Loudness: Loudness is a subjective attribute that depends on the sound wave intensity and frequency. Individuals may perceive the loudness of the same voice differently. The loudness is measured in decibels (dB) which is a measure of intensity. Moreover, frequency has a significant effect on how loud a sound seems. Since the ear is more sensitive to some frequency ranges, i.e., 2000 to 5000 Hz, the same intensity is perceived as louder than the sounds near the high and low-frequency extremes. Dynamics is also a general term in music that correlates with loudness and is concerned with how loud or quiet

different notes should be [76].

3. Timbre: Timbre or tone color is a subjective quality of musical sound that helps us differentiate between two instruments or voices when they produce the same pitch with the same loudness. Different instruments generate different harmonics of a fundamental frequency with varying intensities and therefore generate different timbres.

An audio signal is an electrical representation of the sound wave, where air pressure variations are converted into an analog electrical signal by a transducer such as a microphone. In other words, the instantaneous air pressure is presented as the instantaneous voltage or current. The electrical signal can also be recorded by a recording machine, such as a tape recorder into a magnetic tape or a record cutter into a vinyl.

Digital audio is obtained when the analog audio signal is converted into a digital format using an Analog-to-Digital Converter (ADC). This technology is a more accessible and cheaper way to record, manipulate, and reproduce sound. Since the analog audio is a continuous signal, an ADC discretizes the signal in terms of time and amplitude. Thus, the sample rate and the bit depth are two parameters that need to be considered in digitization. The sample rate determines the number of times we sample one second of audio. The bit depth indicates how many bits are

used to quantize the amplitude.

Consider an audio file with a sample rate of 44.1 kHz and a bit depth of 16 bits per sample (bps). Every second of this audio signal has been sampled 44100 times, and every sample is quantized using 16 bits encoding 2^{16} different values for the amplitude. The sample rate of digital audio files mainly ranges from 16 kHz to 48 kHz to satisfy the minimum sample rate based on the Nyquist–Shannon sampling theorem concerning the analog audio reconstruction performed by a Digital-to-Analog Converter (DAC).

While reproducing and playing audio, the number of communication channels the signal passes through indicates if the sound is monophonic (one audio channel) or stereophonic (two audio channels).

It is clear that from a sample-based perspective, audio signals have high temporal resolutions, and even ten seconds of audio seems like a massive amount of data. Accordingly, we sometimes need to resample the audio with lower rates or quantize the audio using fewer bits for raw audio processing. Mu-law encoding or other companding algorithms are usually used for quantization, and their application is twofold. Firstly we reduce the number of bits for the sake of less computational complexity. Secondly, mu-law encodes the audio in a semi-logarithmic manner so that encoding sensitivity toward the softer parts of the signal is higher, which is similar to the human perception. Applying 8-bit mu-law allows quantizing the

amplitude into 256 different values as follows:

$$F(x) = sgn(x)\frac{\ln(1+\mu\,|x|)}{\ln(1+\mu)} \tag{2.1}$$

where $-1 \leq x \leq 1$ and $\mu = 256$.

2.2.1.2 Time-frequency Representation

Time-frequency representations are obtained by applying the Fourier transform and accounts for audio at frame-level rather than sample-level. The Fourier transform captures the frequency and phase content of the audio signal. As mentioned earlier, every audio signal embodies a series of frequencies, and decomposing a signal into these frequencies provides a good representation of that signal. In order to account for the signal changes over time, short-time Fourier transform (STFT) is used, which applies fast Fourier transform to windowed segments of the audio. The result will be a three-dimensional representation called a spectrogram in which one axis accounts for time, one accounts for frequency bins, and one represents the loudness of frequencies at every time step usually reported in decibels and shown by the color intensity.

Human audio perception can distinguish lower frequencies better than the higher ones and has been modeled experimentally based on this property, resulting in a

logarithmic scale known as Mel-scale. Mel-spectrograms replace the linear or log spectrograms in many studies so that the frequencies are presented according to the Mel-scale and close to our auditory perception. In that regard, many researchers find the Constant-Q Transform (CQT) a better substitute for STFT as it exhibits higher spectral resolution for lower frequencies and higher temporal resolution for higher frequencies.

Time-frequency representations are also helpful in extracting a variety of audio features, one of which is the Mel frequency cepstrum coefficients (MFCCs). In simple words, MFCCs contain information about the vocal tract of speech sources and timbre-related information of musical instruments. MFCCs are obtained by applying cosine transform to the log of the Mel-scaled spectrum of the audio signal.

It should be noted that for the sake of simplicity, the phase contents are often ignored when processing the audio in the frequency domain. However, there are times that the signal should be converted back to temporal audio. In such cases, the Griffin-Lim algorithm is used to convert STFT representation to waveform domain by approximating the phase data, which we cannot consider as a loss-less reconstruction.

2.2.1.3 Symbolic Representation

Symbolic representations provide a high-level description of musical data that is simply understandable by humans and can be easily parsed by computers. Piano-rolls and MIDI are the most commonly used symbolic representations.

During the 19th and 20th centuries, continuous rolls of paper called piano rolls were used by self-playing pianos. Each music note took a whole line in the roll, and the places where the note should be played were punched with its length dependent on the note duration. The same idea is being used in the piano-roll representation. It is a two-dimensional visualization of notes showing the pitch values on one axis and the time-related information such as the start and end of the notes on another axis.

Musical Instrument Digital Interface (MIDI) is a communication protocol that enables electronic musical instruments, audio devices, and music software programs like Digital Audio Workstations (DAW) to connect for editing and recording purposes. As a symbolic music representation, MIDI embodies messages that encode musical activities such as the pitch, velocity, duration, and onset.

Many studies music use data in the form of piano rolls and MIDI to train their models to find patterns among the events or generate the output in these formats. Although using symbolic representations is more straightforward than other rep-

resentation techniques, they cannot represent detailed information on human-like performance or human voice.

2.2.2 Audio Synthesis

Audio synthesis is the task of generating sound using computational techniques controlled by user-defined settings and was traditionally performed using physical modeling and acoustic modeling. In physical modeling, mathematical approaches are employed to create the waveform, while acoustic modeling applies oscillators and filters to produce an acoustic wave with desirable attributes. Nonetheless, these methods provide limited control over the output. [51]

Today, deep generative models have reached the ability to learn the underlying distribution of audio features seen in the training set and generate novel audio samples with respect to the learned distribution. These models allow exercising control parameters over the output to use them as robust speech vocoders or to generate musical sounds conditionally. The audio synthesis systems are mainly categorized into two classes based on how they model the latent structure: autoregressive models and GAN-based models, which will be discussed below.

2.2.2.1 Autoregressive Models

We briefly mentioned the autoregressive nature of some generative models like PixelCNN [78] applied to image synthesis before. They model the joint probability of every pixel of an image as the product of conditional probabilities of the preceding pixels in that image. WaveNet [77], one of the recent groundbreaking audio synthesis models, leverages the same idea in which every audio sample is predicted using the product of the preceding audio samples' probability as shown in equation below:

$$p(x) = \prod_{t=1}^{T} p(x_t | x_1, ..., x_{t-1}) \tag{2.2}$$

where $x = \{x_1, x_2, ..., x_T\}$ is the audio signal.

WaveNet uses one-dimensional convolutions to process audio samples. Nevertheless, there are two key factors that need to be considered in the design of this model. The first is that the network should not have access to future data samples at every step. PixelCNN takes advantage of masking convolutions to remove the impact of future pixels. In WaveNet, this requirement has led to the utilization of causal convolutions to keep future data out of reach by shifting methods. Second, we know that raw audio enjoys a high temporal resolution, and reaching an extended receptive field requires an inapplicably large number of layers. WaveNet

addresses this issue by introducing dilated convolutions that skip a number of samples according to the dilation factor, and as a result, the receptive field increases exponentially in every layer.

Another trick that WaveNet has borrowed from PixelCNN is to apply gated activation units. When the first version of PixelCNN was proposed, although it was faster, it could not beat PixelRNN in terms of performance. One of the reasons is believed to be the gated activations in the LSTM architecture which have been proven effective in PixelCNN and WaveNet as well. Sigmoid and Tanh nonlinearities are adopted in the formation of the gated activation unit as follows:

$$z = tanh(W_{f,k} * x) \odot \sigma(W_{g,k} * x) \tag{2.3}$$

where $*$ and $\odot$ represents convolution and element-wise multiplication operators respectively, $W_{*,k}$ is a learnable convolution filter, k is the layer index, and f and g denote filter and gate.

WaveNet has also provided the opportunity of conditional synthesis by taking a condition vector as an input that determines some of the characteristics of the target audio. It is a great advantage for multi-speaker speech synthesis, where the speaker's identity specifying the voice timbre, accent, and tone is provided as a condition. In text-to-speech synthesis, linguistic embeddings could be applied as

the condition. In speech or music synthesis, the spectrogram can be presented as the condition, and the network can successfully replace the imperfect Griffin-Lim algorithm when processing time-frequency representations. If gated activation unit and the conditioning feature are put together, the activation function will become:

$$z = tanh(W_{f,k} * x + V_{f,k}^T h) \odot \sigma(W_{g,k} * x + V_{g,k}^T h) \tag{2.4}$$

where h is the condition and $V_{*,k}$ is a learnable linear projection.

Engel et al. [31] designed and implemented a WaveNet autoencoder for generating musical notes. In their model, the WaveNet-like encoder with non-causal dilated convolutions outputs a temporal embedding, and the WaveNet decoder generates the audio conditionally. According to the authors, their motivation was to remove the need for an external condition as the encoder learns to provide the required condition in the embedding. They collected a dataset of musical notes called NSynth and used it for training the autoencoder. The model's encoder is able to capture the notes' attributes such as pitch and timbre and provide a strong signal to the decoder that cannot be ignored. [31]

The Wavenet-based architectures that model the raw audio directly have the privilege of accounting for phase information and show higher fidelity compared to phase reconstruction algorithms.

SampleRNN [73] is another autoregressive model aiming at unconditional audio synthesis. GRUs or LSTMs handle the sequential modeling of the probabilities in this architecture. In order to mitigate the complication of long-term dependencies of audio, SampleRNN adopts a hierarchical structure with multiple tiers. Each of the tiers accounts for a different temporal resolution. Higher tiers address lower temporal resolution by operating on non-overlapping frames and condition the high-resolution tiers below them, and finally, the lowest tier nodes have the required information to model the probability of every sample using softmax function. SampleRNN's advantage over WaveNet is its flexibility in allocating computations resources since not all the tiers require the same resources as those needed in the sample-level tier.

2.2.2.2 GAN-based Models

Following the success of GANs in image generation, it was expected that by replacing images with spectrograms (as an image-like representation of the audio), GANs could be applied to audio generation. However, using lossy estimations like the Griffin-Lim algorithm does lead to undesirably noisy outputs.

WaveGAN [26] was the first successful GAN-based audio generator that has been built based on DCGAN and processes raw audio using one-dimensional convolutions. The generator exploits transposed convolution to increase the receptive

field and can produce one second of audio at 16 kHz. A method called phase shuffling is used to stop the discriminator from allowing artifactual patterns caused by transposed convolutions.

In contrast to WaveGAN, GANSynth [30] with a convolutional architecture operates on a time-frequency representation that encodes both magnitude and phase. The network learns to capture timbre-related information by global conditioning on pitch information. The presence of phase information boosts the audio quality compared to WaveGAN but not to the extent that it beats autoregressive models.

MelGAN [63] is a raw waveform generation model designed for text-to-speech and music synthesis. It mainly functions as a mel-spectrogram inversion system. The generator has a fully convolutional architecture. It exploits residual blocks and dilated convolutions to increase the receptive field. There are three discriminators in the multi-scale architecture of MelGAN. Each discriminator operates on different audio scales to model the audio at different resolutions.

GANs are faster than autoregressive models both in the training and inference phase. GAN-based models' number of learnable parameters is considerably lower than autoregressive models, making GANs computationally more efficient. Nevertheless, they have not been able to surpass autoregressive models in terms of fidelity.

2.2.3 Music Source Separation

Audio source separation is the task of isolating the different sounds of a mixture signal. A famous example of source separation is the cocktail party problem in which people are talking simultaneously in a room, and the listener tries to follow one of the discussions. Although the human brain can isolate one of the speech sources, it is a complex problem in digital signal processing.

In the music domain, source separation refers to recovering the contributions from different stems. Stems are fragments of a whole music piece, packaging the tracks of the same instruments. For instance, a piece may include guitar, bass, drums, and vocals stems, where every stem consists of one or more tracks of its corresponding instrument.

Having the stems of a music piece can be useful for remixing and editing purposes and can make some entertainment opportunities like karaoke possible. Music source separation can also assist other music-related tasks such as music transcription, singer identification, and instrument recognition.

The complexity of source separation depends on several factors. For instance, after adding audio effects and mixing the stems, separating the sources will be more difficult, especially when multiple instruments play in harmony. The number of audio channels may impact the level of difficulty. When there are more than

one channels, some source signals can be easily separated depending on the spatial positioning of the source. [11]

Before deep learning approaches were widely used, source separation systems were implemented using different model-based techniques such as kernel, spectrogram factorization, and sinusoidal models. [11] These models separate each of the sources by computing a corresponding binary or soft mask, and by applying the mask to the mixture, that specific source will be extracted. Non-negative matrix factorization (NMF) is one of the most common models used in source separation. NMF factorizes a non-negative matrix M into two non-negative matrices, W and H, by solving a minimization problem. For source separation purposes, NMF is applied to the magnitude spectrogram of the audio represented by M; the values in W represent the spectral characteristic of the sources, and H determines the time activation of those characteristics.

The model-based techniques all have shortcomings and cannot perform well unless the input signal satisfies their conditions. Some of them work better for instrument separation rather than singing voice separation and others operate based on assumptions that are hard to generalize. Fortunately, deep neural networks do not suffer from such drawbacks. Different architectures of neural networks have been investigated in this area.

DeepConvSep [10] with a convolutional encoder-decoder architecture is a spectro-

gram-based model that predicts a soft mask for each source. The magnitude spectrogram of each source is extracted by applying the corresponding mask to the mixture, and using the mixture phase information, the separated waveforms are estimated.

OpenUnmix [97] is a popular spectrogram-based model decomposing music recordings into four stems: bass, drums, vocals, and other instruments. The encoder consists of a 3-layer BiLSTM, and the decoder is implemented as fully connected layers. The model estimates full-band masks to filter the mixture spectrogram.

The before-mentioned models all use spectrograms as input, and we know the phase information does not take part in processing spectrograms. At the time of converting the separated sources' spectrograms into the temporal domain, these models either use phase reconstruction algorithms or use the mixture phase for all the separated sources, both of which are inaccurate. Only recently, advances in deep learning have enabled us to process raw audio easier, and as a result, studies on sample-based source separation are being conducted.

Wave-U-Net [96] is one of the early successful models operating in the time domain for multi-instrument and singing voice separation inspired by the U-Net [84] architecture. It handles long-term dependencies of audio by learning feature maps through the down-sampling path of the encoder using 1D convolutions. The up-sampling blocks at each layer apply linear interpolation to the low-resolution

feature maps, and by combining them with the encoder feature map of the same layer, the network predicts the samples of each source directly.

Demucs [22] is a state-of-the-art waveform-based model with a convolutional autoencoder architecture. It also employs skip connections between the encoder and decoder similar to U-Net and Wave-U-Net. The encoder downsamples the audio using 1D convolutions followed by a BiLSTM network at the bottleneck layer. The decoder employs transposed convolution for upsampling and predicts the source samples separately inspired by the music notes synthesis model proposed in [23].

Although music source separation systems have come a long way, they still have limitations that need to be resolved. The existing models should be trained on isolated sources; they can only separate the sources they are trained for with the number of sources known in advance. Due to copyright restrictions, such a dataset is hardly available for free. The largest free dataset available is MuseDB [82] with 10 hours of data only containing the stems of bass, drums, vocals, piano, and other accompaniment. At this point, separating all the instrument stems is impractical, and those we can separate have artifacts as inevitable parts of them.

2.2.4 Music Style Translation

We previously discussed how image-to-image translation emerged to automate cross-domain mappings of image styles. In recent years, translation techniques have been

investigated in the audio domain, where we encounter two main streams: voice conversion and music style transfer.

Voice conversion, both for speech and singing vocals, deals with translating the voice from a source speaker to a target speaker by changing the speaker's identity attributes like the timbre, accent, and emotion, uttering the same content. Voice conversion applications are seen in personalized speech synthesis and voice dubbing for movies.

Before deep learning methods emerged, voice conversion was studied through statistical approaches like Gaussian mixture models and vector quantization. [94] These models, along with the earlier translation networks, were designed based on parallel data. Such networks require paired data for training and also take two inputs, one representing the content and another representing the voice type. In numerous studies around voice conversion with parallel data, fully connected networks, LSTMs, and encoder-decoder architectures have been employed. In recent years, the advent of image style translation using GAN-based models has made unpaired voice conversion possible. For instance, the CycleGAN architecture has been used in multiple studies leading to multiple versions of CycleGAN-VC [56, 57, 58].

As our research is concerned with music style translation, we will discuss the studies in this area in more detail. Although it is an inaccurate terminology, music style transfer and music style translation are used interchangeably in the literature.

Music style transfer, similar to its counterpart in the image domain, works with paired data to combine the content of one with the style of the other, whereas style translation is not limited to this setting.

Music style translation takes various forms due to various definitions of style in music. Dai et al. [20] classify music style translation into three categories:

1. Composition style translation: changing those features of a music piece addressed in reharmonization while keeping the piece's structure unchanged.

2. Timbre style translation: changing the timbre of a music piece from one instrument or source of style, such as whistling, to another.

3. Performance style translation: changing the performance-related features from one artist to another artist in terms of their expressiveness.

Recent studies have mainly focused on timbre and composition styles. Almost all the composition style translation systems deal with symbolic representations, i.e., MIDI and piano rolls. On the contrary, timbre translation systems mostly use time-frequency representations.

The following paragraphs provide details on the existing composition style translation models.

The composition style translation system in [71] addresses homophonic music composed of a predominant melody and accompaniments. The system uses the

melody as the condition to translate the accompaniment into jazz or classical style using piano roll matrices. The network architecture is inspired by the LSTM-based music generation model called DeepBach [40] to account for temporal dependencies. It also employs dilated convolution blocks similar to WaveNet to model the joint probability of pitch information.

Hung et al. [50] propose two composition translation models that mainly focus on rearrangement. Instead of a direct transfer, the audio's CQT representation is transcribed into a symbolic representation, specifying the instruments and pitch content. Following the style transfer setting, the networks combine the pitch content of a source clip with the timbre information of a target clip. Both models, DuoED and UnetED, have fully convolutional encoder-decoder architectures and learn pitch and timbre representations separately. DuoED uses two different decoders to separate the representations, while UnetED with a U-Net-like structure applies adversarial training to disentangle pitch and timbre information.

In contrast to other models that overlook the dynamics, MIDI-VAE [9] accounts for note velocities. MIDI-VAE is a GRU-based variational autoencoder that changes the pitches, note velocities, and instruments of the accompaniment part toward one of the target styles of classical, jazz, or pop. The network consists of three encoders and three decoders to model pitch, instrument, and velocity distributions. The model also employs a style classifier to guarantee the disentanglement of style-

related information in the latent representation.

Similarly, Groove2Groove [17] keeps the main melody of a source piece and transfer the accompaniments into the target style specified by the target piece. The system is an extension to the one introduced in [16] which could only support a limited number of styles and ignored velocity. The network consists of two encoders to capture the content and style implemented by CNNs and GRUs. The decoder adopts an architecture based on sequence-to-sequence models with attention.

Wang et al. [106] have introduced the only GAN-based model that operates on symbolic representation. They consider the unsupervised translation task a domain adaptation problem and tackle it using a bidirectional GAN structure. One GAN translates the source domain to the target domain, and the other does the inverse. This configuration is used to form a cycle consistency loss similar to the one in CycleGAN.

The following paragraphs provide details on the existing timbre translation models.

Modulated variational autoencoders (MoVE) [5] were first introduced to perform multi-domain timbre transfer with only one VAE rather than multiple decoders. The joint and conditional distributions are modeled in the latent space for several instruments by adopting efficient domain conditioning and external control variables, alleviating the need for adversarial training. The model operates on

Mel-spectrograms and converts them back to the audio domain using Griffin-Lim.

The timbre translation model in [70] is inspired by the MUNIT approach and uses a Relativistic average GAN (RaGAN) [55] to translate the timbre of the input into one of the three instruments of piano, guitar, and string quartet. The network operates on Mel-spectrograms and exploit MFCCs in the consistency loss to ensure the timbre-related features are captured.

TimbreTron [48] is another GAN-based model which applies CycleGAN architecture to the CQT representation of audio. Then a WaveNet synthesizer reconstructs the audio conditioned on the generated spectrogram. TimbreTron operates in the style transfer setting and takes both the source and target timbre as the input.

The first attention-based timbre transfer system was introduced in [54] based on the CycleGAN architecture with an attention-guided generator. This architecture was initially developed for image-to-image translation. Therefore, Mel-spectrograms are utilized as an image-like representation of the audio. The system employs a MelGAN conditioned on the generated Mel-spectrograms to reconstruct the audio.

The very recent work by Cífka et al. [15] presents an extension of the vector-quantized variational autoencoder (VQ-VAE) operating via disentangled pitch and timbre representations. VQ-VAE is a variational autoencoder with discrete latent

representation, which is used for modeling the pitch content. A regular encoder is also employed to produce the style code. In order to ensure style and content disentanglement, using paired data is avoided in the training phase. Both encoders and the decoder are composed of convolutional layers followed by GRU layers. The model uses log-STFT representation, and Griffin-Lim is used for audio reconstruction.

The universal timbre translation network in [75] is the only model to the best of our knowledge that processes raw audio waveforms. Inspired by the WaveNet autoencoder architecture in [31], it translates an arbitrary source piece to several specific timbre domains. Although one universal encoder is used to code the pitch-related features, every domain needs to have a specific conditional WaveNet decoder. The decoders are conditioned on the pitch content and generate audio that entails the specific timbre they have been trained for. A convolutional confusion network is adopted to ensure that no timbre-related information is encoded in the latent representation so that the decoders cannot memorize the pitch information.

Table 2.1 shows a summary of the models addressing music style translation with some of their attributes.

So far, we have discussed all the network architectures, models, and studies that will assist in better understanding the contributions of this thesis, which will be elaborated on in the following chapter. Inspired by the work in [75], we propose

Table 2.1: Music style translation models and their attributes.

Category	Reference	Styles	Input Domain	Architecture
Composition Style Translation	[71]	Jazz, Classical	Piano-roll	LSTM + WaveNet
	[50]	Instrumentation (Piano, Strings, Acoustic, Band)	MIDI	Encoder-decoder
	[9]	Jazz, Pop, Classical	Piano-roll	VAE
	[16], [17]	Jazz, Pop, Instrumentation (Piano, Guitar, Bass, Drums, Strings)	MIDI	Encoder-decoder
	[106]	Jazz, Pop, Classical	MIDI	GAN
Timbre Style Translation	[5]	Saxophone, Flute, Violin, French-Horn	Mel-spectrogram	VAE
	[70]	Piano, Guitar, Strings	Mel-spectrogram	RaGAN
	[48]	Piano, Flute, Violin, Harpsichord	CQT	CycleGAN + WaveNet
	[54]	Piano, Violin	Mel-spectrogram	CycleGAN
	[15]	Keyboards, Guitars, Bass, Woodwinds, Strings	Log-STFT	VQ-VAE
	[75]	Piano, Strings, Woodwinds	Raw Audio	WaveNet Autoencoder

a model for audio-based multi-instrument music translation. The ultimate goal of our research is to perform automatic music rearrangement by altering the set of performing instruments, which was first addressed in [50]. To this end, we leverage the WaveNet autoencoder architecture [31] comprising convolutional encoders and WaveNet decoders. The universal timbre translation model [75] and the state-of-the-art audio source separation model [22] will also be employed in our experimental setups.

3 Methodology

In this chapter, we introduce Music-STAR, a model based on WaveNet autoencoders specifically designed for audio-based multi-instrument music translation that promotes audio-based rearrangement. We propose Music-STAR with two training approaches, unsupervised and supervised, both of which will be elaborated in Section 3.4.

Before delving into the details of our proposed model, we present the StarNet dataset, used to train the architectures forming our research experiments in Section 3.1. Next, we present two baseline models to perform multi-instrument translation: single-instrument translation (Section 3.2), and separation-translation pipeline (Section 3.3), which will be compared to Music-STAR in terms of translation quality.

3.1 Dataset

Having the right dataset in terms of quality and quantity is the first step towards successfully training deep architectures. Conducting this thesis requires a dataset containing multi-track pieces played with different instruments alongside their stems. Since accessing actual songs' stems is hardly possible due to copyright limitations, we use a multi-track dataset in MIDI format and then convert it to audio by applying virtual software instruments.

Virtual software instruments are pieces of software that translate symbolic music notation, such as MIDI, to audio. Different techniques are used to create such instruments, such as sampling and using synthesizers. Today, almost all real instruments are available as a virtual instrument, and it is trivial to convert MIDI inputs into audio performed by arbitrary musical instruments using a DAW.

In this manner, we create StarNet, a dataset for training and evaluating MusicSTAR, and our baseline models. In StarNet, every piece consists of two instruments, one accounting for the melody line and the other complementing the melody as the accompaniment. The MIDI files we used are mainly from the MusicNet dataset [100]. MusicNet is a collection of freely licensed classical music recordings containing the work of 10 well-known classical composers and 11 instruments in total. We applied virtual software instruments using GarageBand (Figure 3.1), a digital audio

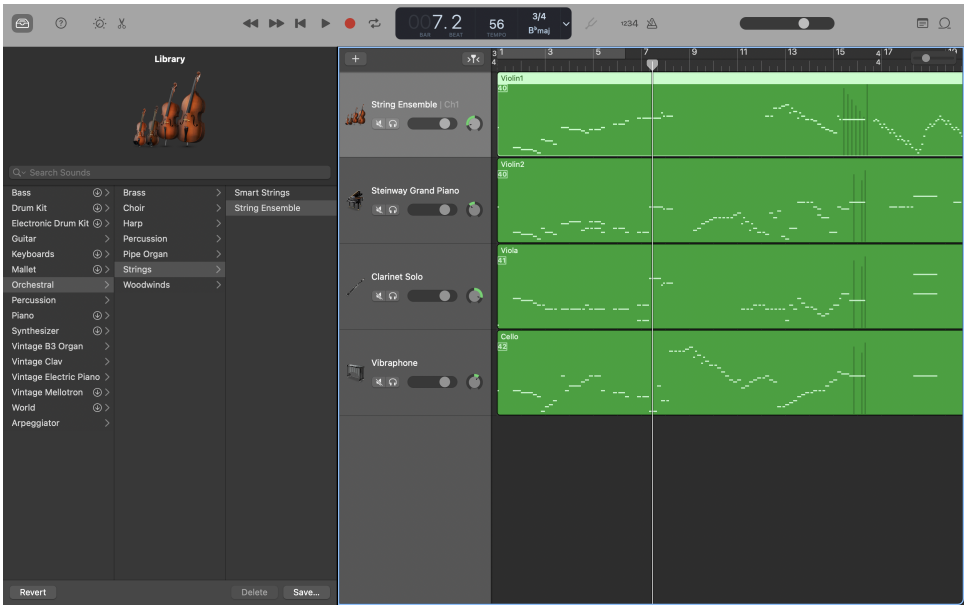

Figure 3.1: GarageBand Environment - The library section on the left is a collection of available virtual instruments. Each of the instruments in our dataset has been assigned to one of the MIDI tracks.

workstation developed by Apple.

We previously argued that one advantage of raw audio over MIDI representation is incorporating performance details, determined by articulations, dynamics, and the performer's choice of expressiveness. Since StarNet is created by applying virtual instruments to MIDI data, it is not necessarily rich in such details either. Nonetheless, StarNet is sufficient to exhibit the capacity of Music-STAR as an audio-based rearrangement model. More advanced ingredients can be investigated using a more ornate dataset in the future.

Figure 3.2 shows the process of applying virtual instruments to MIDI tracks and

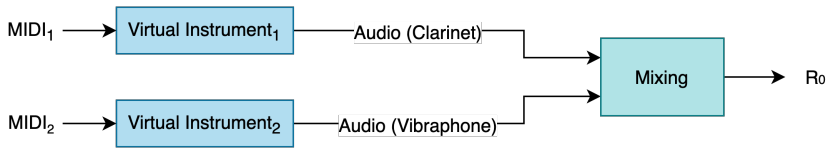

Figure 3.2: The process of obtaining the target arrangement by applying virtual instruments to MIDI input.

mixing them into a two-instrument piece. Through this approach, we also obtain the gold standard for our experiments (R_0), which indicates how the rearranged outputs should sound ideally according to the quality of pieces in our dataset.

We present three versions of StarNet, raw, pre-processed, and reduced, the details of which are discussed below.

3.1.1 Raw Data

We utilize four virtual instruments in StarNet, piano, string ensemble (including violin, viola, cello, and bass accounting for different pitch ranges), vibraphone and clarinet.

The dataset is composed of two domains created by strings-piano and clarinet-vibraphone combinations of one piece. Six audio files present every music piece in StarNet:

- First MIDI track performed by strings.
- Second MIDI track performed by piano.

– Strings-piano mixture.

– First MIDI track performed by clarinet.

– Second MIDI track performed by vibraphone.

– Clarinet-vibraphone mixture.

so that strings and clarinet can be the source/target instruments for the first track, while piano and vibraphone determine the source/target instruments for the second track.

We have chosen piano $\leftrightarrow$ vibraphone and strings $\leftrightarrow$ clarinet translations because we want the source and target instruments to be controlled by the same type of excitation while being played. More precisely, piano and vibraphone notes originate with discrete excitations, while clarinet and strings notes can be played with a continuous stream of excitations.

As mentioned before, StarNet is created based on the MIDI files from MusicNet, which contains solo performances, duets, and a few musical ensembles. As the tracks are separated in the MusicNet MIDI files, converting them to audio both separately and as a mixture is possible.

We select a variety of music pieces from the MusicNet collection, and instead of using their original instruments, we apply the above-mentioned virtual instruments. Some of the pieces are originally composed for solo piano, and we apply two different

instruments to the right-hand and the left-hand tracks. For instance, we keep the right-hand track performing the melody with the piano while applying the string ensemble to the left-hand. In other cases, we pick two tracks of a string quartet, accompanied clarinet pieces, or other woodwind instruments ensembles. The resulting dataset contains 100 two-instrument classical pieces adding up to roughly 11 hours. We render the audio as stereo WAV files with a sample rate of 44.1 kHz and a bit depth of 16 bps.

3.1.2 Pre-processed Data

After creating the raw dataset, we pre-process it using the functions explained below.

1. Audio Trimming: Since the instrument tracks and their combination are used to train the models at sample level, it is necessary to make sure that all the tracks of one music piece have the same length, i.e., the same number of samples. Otherwise, the alignment between the individual tracks and the mixture could be disrupted. There are many pieces in the raw dataset that have one instrument track longer than the other resulting in two audio files with different durations. To mitigate this issue, we apply audio trimming to discard the extra samples at the end of the longer tracks.

2. Silence removal: In the raw version of StarNet, there are inevitably many intervals in every piece that one or both instruments are silent. However, to deliver the purpose of our proposed model, we should make sure that the two instruments are played simultaneously for a considerable amount of time and reduce the duration of solo parts as much as possible. Therefore, we need to detect the silent intervals by indicating the minimum loudness and minimum silence duration and remove all those intervals from all tracks of a particular piece. After the silence removal, the dataset size shrank to roughly 9 hours.

3.1.3 Reduced Data

The pre-processed version of StarNet includes uncompressed stereo WAV files with a sample rate of 44.1 kHz and depth of 16 bps, i.e., the audio quality of a CD. We can reduce the amount of data by resampling the audio at a lower sample rate. We resample the audio at a rate of 16 kHz, which results in a lower quality than 44.1 kHz, but with enough resolution to present all different timbres.

Moreover, using stereo audio files doubles the volume of data and the number of parameters in the models accordingly. In order to reduce the data size, we merge the two audio channels and output mono audio.

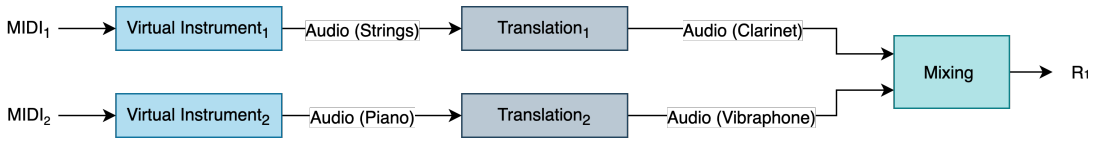

Figure 3.3: The process of obtaining the target arrangement by applying single-instrument translation.

3.2 Single-Instrument Translation

The most simplistic approach toward multi-instrument translation is applying single-instrument translation to each instrument. However, this is only possible when the music stems are available. We consider this approach as a baseline in our experiments and will evaluate how Music-STAR performs compared to this baseline. Figure 3.3 shows how a new arrangement (R_1) is generated by applying single-instrument translation and mixing the translated tracks. The obtained results from this task are expected to be of lower quality than R_0 in Figure 3.2.

Since we discuss audio-based rearrangement in this thesis, we employed the universal network [75], the only audio-based music translation model, which is built upon the WaveNet autoencoder architecture [31] trained in an unsupervised manner. This model consists of a universal encoder and six decoders corresponding to six domains used for network training, all collected from the MusicNet audio dataset:

– Bach's solo cello

– Beethoven's solo piano

– Cambini's woodwind quintet

– Bach's Solo Piano

– Beethoven's accompanied violin

– Beethoven's string quartet

The temporal encoder with a WaveNet-like architecture learns the pitch content embeddings, while for every target domain, an individual decoder is trained to generate realistic audio by being conditioned on the encoder's embedding. In order to prevent the autoencoder from memorizing the data and make it capture meaningful features in the code, a distorted version of the input is fed into the network. In other words, the network is trained as a denoising autoencoder that learns to recover the original input.

In addition to the encoder and the decoders, a domain classification network is employed for adversarial training of the encoder using the domain confusion loss [34]. This confusion network is used to prevent the encoder from encoding the input's domain-specific features. Accordingly, the confusion network should not be able to classify the generated embeddings into the correct domain.

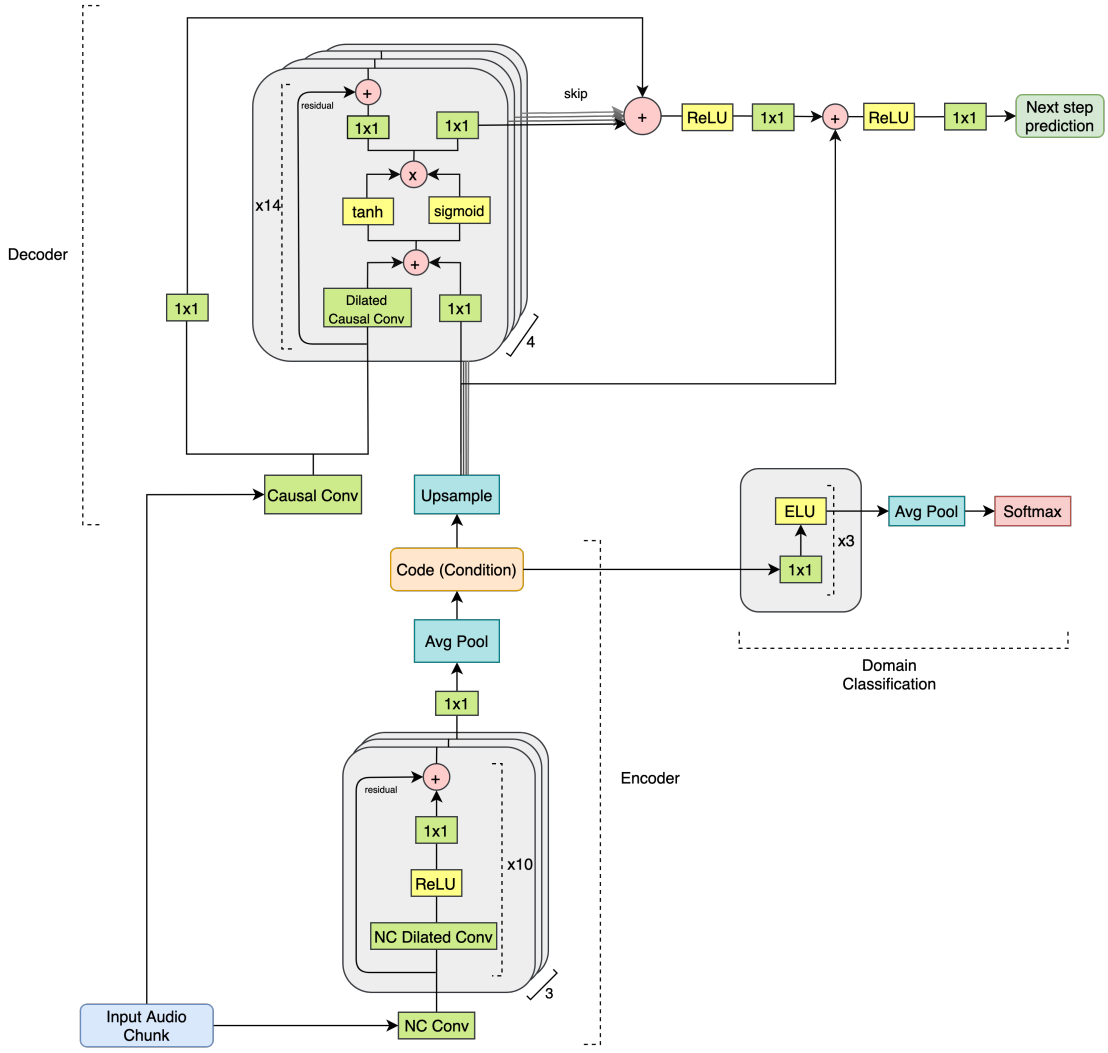

Figure 3.4: The detailed view of the universal music translation network [75] architecture. The confusion network and audio input of the decoder are employed only during training.

3.2.1 Model Architecture

The detailed view of the network components is shown in Figure 3.4. The architecture is almost identical to the original WaveNet-autoencoder presented in [31]. The encoder starts with a non-causal non-dilated convolution followed by three blocks of 10-layer residual networks composed of non-causal dilated convolutions with a kernel size of 3 followed by ReLU nonlinearity and a 1x1 convolution. The dilation increases in every layer by a factor of 2, while the channel number, i.e., 128, remains unchanged throughout the layers. The residual summation of every layer creates the input for the next layer. The final code of size 64 is obtained by applying a 1x1 convolution and average pooling to the output of the final layer.

The code is then upsampled using nearest neighbor interpolations and is fed into the decoder as the condition. The WaveNet decoders are implemented with four blocks each consisting of 14 layers. A causal dilated convolution with a kernel size of 2 is applied to the input, and a 1x1 convolution is applied to the condition in every layer. Sigmoid and Tanh nonlinearities are then used as gated activation units applied to the sum of the input and the condition. Residual connections provide the input for the subsequent layers, while skip connections pass the output of every block. The final output will be the result of 1x1 convolutions followed by ReLU activations applied to the summation of the skip connections. Since 8-bit

mu-law encoding is applied to the input, the final layer of the decoder predicts every sample with 256 possible values.

The confusion network used for adversarial training consists of three 1x1 convolution layers with a channel size of 100. It uses ELU nonlinearity and outputs a k-dimensional vector where k is the number of domains. If the encoder succeeds at excluding the domain-specific information, it is expected that the confusion network cannot classify the generated code better than random.

3.2.2 Model Training

The network is originally trained on six classical music domains from the MusicNet dataset. Since the available amount of data for different domains varies, the model has a different level of expressiveness in translating to each domain.

In [75], the authors emphasize the role of data augmentation, which is performed as modulating the pitch locally in every training audio segment. While the encoder learns to capture the pitch content of the augmented data, the decoder learns how to denoise the distorted audio. The teacher forcing technique is used for training the decoder where the ground truth of one time-step is applied as the input of the next, enabling the decoder to be trained in parallel. On the contrary, no data augmentation is applied during inference, and the decoder generates the output sequentially due to its autoregressive nature.

Training the network takes place in a jointly manner, in which the encoder, the decoder, and the domain confusion network get updated according to the loss functions they correspond to. The universal encoder and the decoders are trained to minimize the loss below:

$$\sum_{j}\sum_{s^j}\mathbb{E}_r L(D^j(E(O(s^j,r))),s^j) - \lambda L(C(E(O(s^j,r))),j) \tag{3.1}$$

where D, E, O, and C represent the decoder, the encoder, the augmentation process, and the confusion network respectively. L is the cross entropy loss, $j = 1, 2, ..., 6$ accounting for the six domains, and s^j is the input segment which is identical to the target output.

The classification loss of the confusion network is defined as:

$$\sum_{j}\sum_{s^j}\mathbb{E}_r L(C(E(O(s^j,r))),j) \tag{3.2}$$

The original model has been trained on mono audio segments with a sample rate of 16 kHz, that are quantized by 8-bit mu-law. The input files are randomly selected, and then a 0.75-second audio chunk is randomly segmented out of the file for every training input on the fly. A duration between 0.25 to 0.5 seconds of that segment is then selected for applying distortion. Training has been done using Adam optimization with a learning rate of 1e$-$3 and a decay factor of 0.995 with

a batch size of 32. The model was trained for three days on six nodes, each with eight Tesla V100 GPUs, i.e., 48 GPUs in total, for 92 epochs.

In order to use this architecture as a baseline, we needed to make sure that the decoders can generate the instruments' timbre from our datasets which are piano, string ensemble, clarinet, and vibraphone. Therefore, we tried to train the model from scratch using the StarNet pieces performed by the four single instruments. However, emulating such training configuration with a high number of GPUs requires access to considerable resources, which were not available, and thus the generated outputs could not compete with the pre-trained models. As a result, we decided to use the pre-trained universal encoder and finetune the decoders using our dataset. Since the original encoder has been trained on six domains and captures domain-agnostic features, it is powerful enough to encode inputs from other domains successfully.

Furthermore, the domains that the original model has been trained for also include piano, string quartet, and woodwind quintet, which are related to three instruments in the StarNet dataset. We finetuned these decoders using the corresponding instrument pieces from StarNet to ensure the learned timbre is as close as possible to what we are offering in the dataset. We also trained a separate decoder for vibraphone, which was initialized with the pre-trained piano decoder. We finetuned the decoders with a batch size of 16 and a learning rate of 0.98 for 100

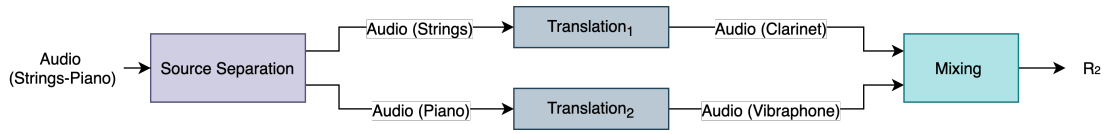

Figure 3.5: The process of obtaining the target arrangement by applying separation-translation pipeline.

epochs using the reduced version of StarNet, where 8-bit mu-law is applied.

3.3 Separation-Translation Pipeline

Although the before-mentioned gold standard and baseline set good examples of what our model's ideal output could be, they do not correspond to actual use cases where the instrument tracks are not available separately, and the system should take the audio mixture as input.

One way to model music rearrangement without having the stems is to adopt a pipeline of audio source separation and single-instrument translation. We design an experiment to demonstrate the performance of this pipeline. The input audio mixtures are from two domains, strings-piano and clarinet-vibraphone, as discussed in the dataset section. The source separation module isolates each of the tracks so that a single-instrument translation model, as discussed in Section 3.2, can transform each of them into a target instrument (Figure 3.5).

To perform source separation we adopted Demucs [22], a state-of-the-art music source separation model that operates at the sample level, taking an audio mixture

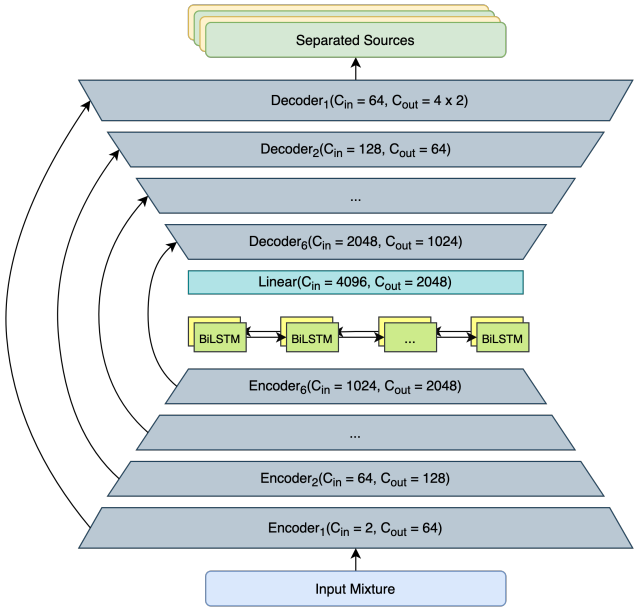

Figure 3.6: Demucs architecture. [22]

and outputting isolated audio waveforms. According to the authors, their architecture is closer to the audio generation models than the masking models since the final outputs are directly the predicted separated stems.

3.3.1 Model Architecture

Demucs has a U-Net architecture with a convolutional encoder and decoder. Skip connections are used to apply residual learning between the corresponding layers of the encoder and the decoder in the downsampling and upsampling paths, respectively.

Figure 3.6 depicts Demucs's architecture. The models' encoder comprises six

convolutional blocks. The encoder takes a stereo mixture, i.e., a two-channel input, and through the first convolutional block, the number of output channels will reach 64. In the subsequent blocks, the number of channels will be doubled per block. Every block consists of a 1D convolutional layer with a kernel size of 8 and a stride of 4, followed by a ReLU function. The feature map is then fed into a 1x1 convolutional followed by a Gated Linear Unit (GLU) [21], which improves performance based on the results of the ablation study. A 2-layer bidirectional LSTM is placed right after the encoder to help with the loss reduction, followed by a linear layer to adjust the size of the final code.

Demucs's decoder has the same number of blocks as its encoder, and every decoder block is almost the inverse of an encoder block except that the 1x1 convolution is replaced by 1D convolution with a kernel size of 3 and stride of 1. This convolutional layer is used to engage the context of neighboring samples while upsampling the code with GLU activation coming afterwards. The upsampling is performed through the transposed convolutions with a kernel size of 8 and a stride of 4, and a RELU activation. Every block of the decoder reduces the number of channels by the factor of 2 to the point that the final output estimates the two-channel (stereo) sources. Figure 3.7 provides a detailed view of Demucs's encoder and decoder layers.

Inspired by Wave-U-Net [96], the first successful audio source separation model,

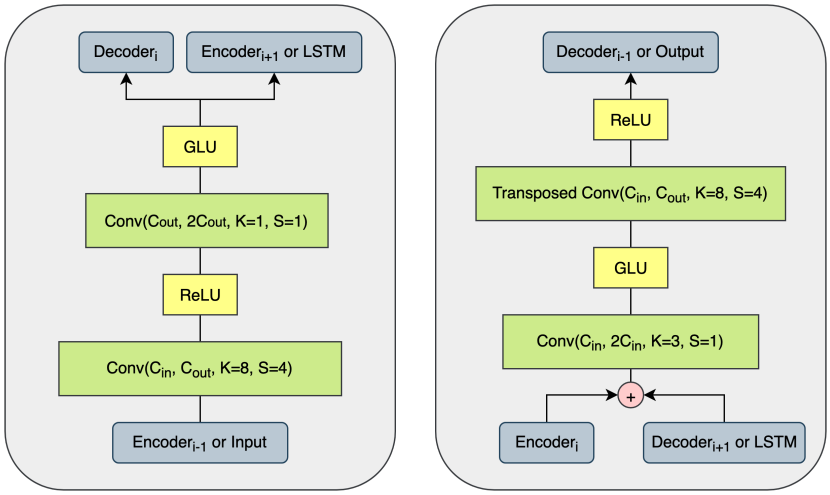

Figure 3.7: Detailed view of the Demucs layers encoders on the left and decoders on the right. [22]

Demucs adopts skip connections to transmit some useful information that is essential for audio reconstruction, such as phase information of the mixture.

3.3.2 Model Training

Demucs is originally trained on MuseDB [82] and is able to separate four stems: bass, drums, vocals, and other accompaniments. As mentioned before, the existing pre-trained source separation models cannot separate every single instrument of the mixture, and it is probably due to the unavailability of all the tracks in the training sets. However, it is vital for our experiment that the network separates the two instruments present in the mixtures. Therefore we trained Demucs using the pre-processed version of StarNet. We randomly selected 80 pieces to use as the

training set and 20 tracks as the validation set. Every mixture audio was segmented into 5-second slices as one instance of the input on the fly, and the corresponding segments of the sources were used as the ground truth. Additionally, four types of data augmentation were applied:

- Shifting: Randomly shifting the audio in time.
- Flipping channels: Swapping the left and the right audio channels.
- Flipping polarity: Multiplying the sources by ± 1 randomly.
- Scaling: Scaling the value of the samples within a fixed range randomly.
- Remixing: Shuffling sources within one batch to generate a new mix.

Demucs is trained to minimize the loss below:

$$\min_{\theta} \sum_{x \in D} \sum_{s=1}^{S} L(g_s(x;\theta), x_s) \tag{3.3}$$

where x is an example from the dataset D, S is the total number of stems, $g_s(x;\theta)$ is the predicted waveform for stem s given x parameterized by θ, x_s is the ground truth for stem s in x, and L is L1-loss:

$$L_1(\hat{x_s}, x_s) = \frac{1}{T} \sum_{t=1}^{T} |\hat{x}_{s,t} - x_{s,t}| \tag{3.4}$$

where x_s is the waveform of stem s containing T samples and $\hat{x}_s$ is the predicted waveform corresponding to stem s.

Demucs was originally trained on sixteen V100 GPUs with 32 GB of RAM using a batch size of 64. We trained the model on eight V100 GPUs using a batch size of 32 for 250 epochs.

We trained Demucs with the original learning rate of 3e$-$4 and Adam optimizer, which were used for training Demucs on MuseDB. The resulting model can take an arbitrary length of an input mixture and successfully isolates strings tracks from piano and clarinet tracks from vibraphone.

3.4 Music-STAR

So far, we have discussed different approaches to account for multiple instruments in music translation:

1. If we have the MIDI tracks, we can apply virtual software instruments.

2. If we have the separated audio sources, we can take advantage of single-instrument translation models.

3. If we have the audio mixtures, we can use source separation models to isolate each instrument track and apply single-instrument translation.

Nevertheless, our goal is to realize the idea of multi-instrument audio rearrangement where we do not face the restrictions of the before-mentioned techniques. To this end, we introduce Music-STAR, a system specially designed for this task. We discuss two approaches: unsupervised and supervised. In the unsupervised approach, there is no need to have paired data, i.e., the same music piece performed in the source and the target domains. On the contrary, having paired data is necessary for supervised training.

The details of these approaches will be unfolded in the subsequent sections. But before that, we introduce a notation of the system's input and outputs to help us present the approaches more precisely.

Every audio track consists of pitch, loudness, duration, and timbre data. To simplify our discussion, we consider the loudness and duration as parts of the pitch-related information. Thus, we can notate a piano audio track as follows:

$$A_{piano} = T_{piano} + P_{piano}$$

where A stands for audio, T is the timbre and P is the pitch. Separating these two components is a study topic known as pitch-timbre disentanglement, which has been leveraged in previous music translation studies [9, 50, 75] and is mostly addressed by adversarial learning in encoder-decoder architectures.

In the universal translation model that we used as our baseline, the autoencoder is originally trained in such a way that the encoder captures the P component, i.e., the pitch content, and the decoder learns to synthesize audio corresponding to the P by applying the timbre T. During the training phase the autoencoder is trained to denoise and reconstruct the input audio. However, any arbitrary input can source the model for pitch information during inference while the decoder applies the learned timbre to the pitch component. For example, violin to piano translation will take place as follows:

1. $Input = A_{violin} = T_{violin} + P_{violin}$
2. The encoder generates a code for P_{violin}.
3. The code is fed into the piano decoder.
4. $Output = A_{v2p} = T_{piano} + P_{violin}$

In our research, we are tackling a more complex problem as we are working with multi-instrument pieces. Our system's input will be either:

- $A_{mix1} = A_{piano} + A_{strings} = T_{piano} + P_{piano} + T_{strings} + P_{strings}$, or
- $A_{mix2} = A_{vibraphone} + A_{clarinet} = T_{vibraphone} + P_{vibraphone} + T_{clarinet} + P_{clarinet}$

As a result, the pitch-timbre disentanglement does not suffice as we need to isolate two timbre and two pitch components. Since we do not intend to engage source separation modules in our system, we propose two methods to deal with such a problem.

3.4.1 Unsupervised Music-STAR

In this approach, we employ a method that does not require paired data from the source and target domains. To be specific, for the input:

$$A_{input} = A_{piano} + A_{strings}$$

the training set does not need to have the corresponding output:

$$A_{output} = A_{vibraphone} + A_{clarinet}$$

where:

$$P_{piano} = P_{vibraphone} \quad \text{and} \quad P_{strings} = P_{clarinet}$$

Nonetheless, what we use in this approach are the stems of the input: A_{piano} and $A_{strings}$.

Unsupervised Music-STAR performs a semi-separation task through the encoding process, where the encoder learns to capture the pitch content of one of the two instruments in the mixture. To this end, we need a supplementary encoder that can provide the pitch content for single instruments and assist the Music-STAR encoder in distinguishing between the instruments. Recall that the universal encoder in our baseline model learns to extract pitch-related information from the input. Thus the output of such an encoder can provide the exact information we want to be extracted from a mixture. Based on this, we can train the Music-STAR encoder to mimic the output of the universal encoder when given a mixture as the input. For instance, if we have the mixture as:

$$A_{mix} = A_{piano} + A_{string}$$

we feed the piano track A_{piano}:

$$A_{piano} = T_{piano} + P_{piano}$$

into the universal encoder, and the output will contain piano track pitch information (P_{piano}). We then input the mixture A_{mix} to the Music-STAR encoder and train it to output the same code (P_{piano}). This way, we can finally have a piano-specific

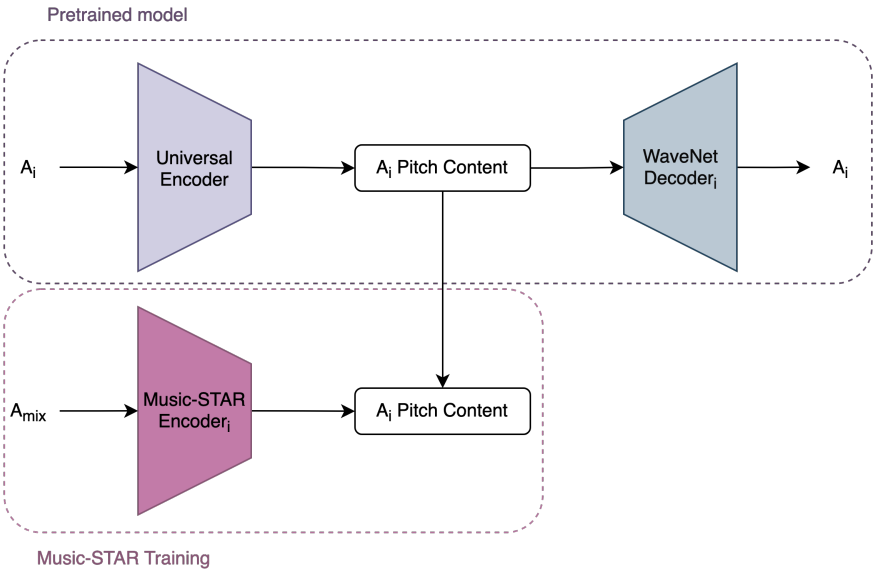

Figure 3.8: Training Music-STAR using the unsupervised approach. The Music-STAR encoder is trained to generate the same code as the universal encoder.

encoder to extract the piano pitch content from every mixture. Figure 3.8 illustrates how this method works and how Music-STAR with unpaired data makes multi-instrument rearrangement possible.

Via this approach, we can isolate the information corresponding to one instrument without applying actual source separation. We can train one encoder per instrument with the help of the universal encoder. During inference, a pre-trained WaveNet decoder can also be engaged to translate the code to an arbitrary target instrument.

Nevertheless, the question may arise: if we are using the components from the single-instrument model to train the multi-instrument model, why not keep prac-

ticing the same idea of the baseline model, in which the instruments are translated one by one and then mixed together? The answer to this question is that we are accounting for the case when we only have audio mixtures as the input. While a tremendous amount of audio data is available on single instruments, our access to multi-instrument pieces along with their stems is quite limited. However, we can mix those single-instrument tracks to generate multi-instrument datasets and train the Music-STAR encoder without concern if they are actual music pieces. Ultimately, the same encoder can isolate one instrument out of many when the input is a true musical mixture.

3.4.1.1 Model Architecture

As explained above, we adopt an autoencoder architecture as the backbone of the model. The focus of this method is to train the Music-STAR encoder with the help of the universal encoder discussed in the baseline model (Section 3.2). The architecture of the universal encoder is depicted in Figure 3.9. Since the Music-STAR encoder should learn to extract the same information as the universal encoder from a mixture rather than a stem, we expect a more complex architecture to benefit the performance. The final architecture of the Music-STAR encoder is shown in Figure 3.9. Similar to the universal encoder, it starts with a non-causal non-dilated convolution. The encoder blocks consist of four blocks of 14-layer residual networks

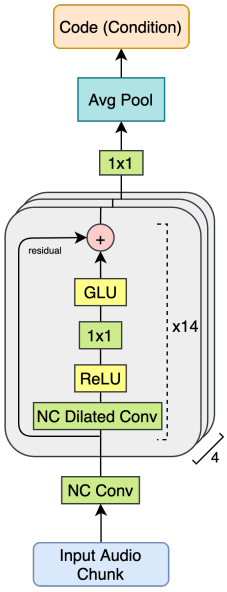

Figure 3.9: Detailed view of the Music-STAR encoder.

composed of non-causal dilated convolutions with a kernel size of 3, followed by ReLU nonlinearity and a 1x1 convolution. A GLU activation function follows the 1x1 convolution, and the output is summed with the input to form the residual connection. The result of this connection is then fed into the next layer of the encoder. The dilation increase factor of 2 and 128 channels are selected identical to the universal encoder. Our experiments show that the presence of the GLU nonlinearity boosts the performance compared to the case where we adopt the same architecture as the universal encoder only with more layers. The formula of

GLU function is as follows:

$$GLU(a, b) = a \otimes \sigma(b) \tag{3.5}$$

where a and b are formed by cutting the input in half along the selected dimension, σ is the sigmoid function, and $\otimes$ is the element-wise product between matrices.

Note that we want the Music-STAR encoder to output a code with the same dimensions as the universal encoder, i.e., 64 channels. Since the GLU activation splits the input, the 1x1 convolution before the GLU doubles the number of the channels. After passing through the GLU, it will be reduced to its original dimension. The average pooling applied to the final code has a kernel size of 800 samples resulting in a code of size 64x15.

The pre-trained WaveNet decoders are attached to the Music-STAR encoder during inference to translate the code into the target instruments, and the summation of their results will demonstrate the rearrangement. The details of the decoder architecture are illustrated in Figure 3.4 and described in Section 3.2.

3.4.1.2 Model Training

Unlike the baseline model that trains a universal encoder for all the domains in the training set, we need to train a single encoder for each instrument to extract

pitch information from a mixture. We feed the instrument track to the universal encoder and use the generated code as the ground truth for training the Music-STAR encoder, which takes a mixture as the input. To be concrete, the model strives to minimize the loss function below:

$$\sum_{j} L(E_m^i(mix_j), E_u(x_j^i)) \tag{3.6}$$

where E_m is the Music-STAR encoder, mix_j is the jth fragment from the audio mixture, E_u is the universal encoder, x_j^i is the jth fragment of the instrument track we want to extract from the mixture, and L is L1 loss.

Similar to the baseline model, the encoder is trained on mono audio segments with a sample rate of 16 kHz from reduced StarNet, that are quantized by 8-bit mu-law. The 0.75-second audio chunks are randomly segmented out of the mixture files, and the same segments are extracted from the instrument tracks. We train the model using Adam optimization with exponential learning rate decay. We adopt a learning rate of 3e$-$4 and a decay factor of 0.98 with a batch size of 16 for a total of 100 epochs.

During inference, what we achieve as the resulting rearranged piece is formulated

as:

$$\sum_{i \in I} D_t^i(E_m^i(mix)) \tag{3.7}$$

where $I = \{1, 2\}$ since there are two instrument tracks in the mixtures and D_t^i is the target decoder chosen for the ith source instrument.

3.4.2 Supervised Music-STAR

This section presents another approach towards audio-based music rearrangement based on the WaveNet autoencoder architecture, where we take advantage of paired audio tracks. In other words, for every mixture consisting of piano and strings

$$A_{mix1} = A_{piano} + A_{strings}$$

there is a counterpart performed by vibraphone and clarinet:

$$A_{mix2} = A_{vibraphone} + A_{clarinet}$$

where $P_{vibraphone} = P_{piano}$ and $P_{clarinet} = P_{strings}$.

Before we extend upon how this method works, we need to illuminate some details from the original WaveNet autoencoder [31] and its enhanced version used

for music translation [75].

Following the WaveNet's success in conditional audio synthesis and speech generation, OpenAI introduced the WaveNet autoencoder to synthesize musical notes with no need for external conditioning. The idea behind this model was to capture the long-term temporal structures by the encoder and then use the generated code to condition the WaveNet decoder to reconstruct the musical notes. Their achievements in manipulation and interpolations of timbre inspired the authors of [75] to take it one step further and use the same architecture for music translation. We look into specific parts of their contribution once again to shed light on how their model helps us tackle our research question in this approach. As previously mentioned in Section 3.2, they trained the WaveNet autoencoder with two critical considerations:

1. Adopting a confusion network ensured that the encoder extracted the pitch information while the decoder learned to apply the timbre-related features.

2. Using a random local pitch modulation, they ensured that the encoder did not memorize the pitch content and trained the whole network to reconstruct the original audio as a denoising autoencoder. To this end, the teacher forcing technique was used in the training phase, where the decoder takes two inputs: the original audio segment and the condition produced by the encoder. On

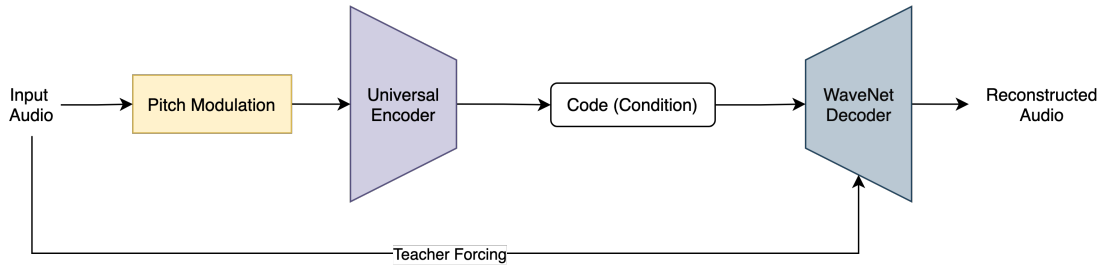

Figure 3.10: The network learns to denoise and reconstruct the input audio segment.

the other hand, the conditioning code is the only input during inference, and the decoder starts from scratch to synthesize the audio sample by sample.

Figure 3.10 presents the model's training procedure. Consider the case when we train the network on a one-second piano fragment A_{piano} where:

$$A_{piano} = T_{piano} + P_{piano}$$

The pitch modulation is applied to a segment of length 0.25 to 0.5 to distort the audio fragment before it is fed into the encoder:

$$A_{in} = T_{piano} + P_{piano} + P_{distortion}$$

Since the encoder learns to extract pitch information, we expect the code to embody $P_{piano} + P_{distortion}$. However, the decoder will learn due to the teacher forcing technique to reconstruct A_{piano}. Given an input of noisy pitch information,

it should make the encoder provide a code for P_{piano} by removing the distortion, and apply timbre T_{piano}. Consequently, the decoder learns to reconstruct the audio correctly, and the encoder learns to generate the proper code that the decoder needs for its task.

Now, we formulate the task of two-instrument translation once again with the input audio as:

$$A_{mix} = A_{piano} + A_{strings} = T_{piano} + P_{piano} + T_{strings} + P_{strings}$$

Our desired output is the rearranged piece in which the piano and string instruments are translated into vibraphone and clarinet, respectively. We expect that a single decoder can only learn the timbre information of one instrument, and we assume that for obtaining A_{out}, we need to mix the results from two separate decoders. To ensure simplicity, we first consider the case where we translate only one of the instruments in the mixture, e.g., piano. Thus the output we are looking for is:

$$A_{out} = A_{vibraphone} = T_{vibraphone} + P_{vibraphone}$$

where $P_{vibraphone} = P_{piano}$.

To achieve this output, we should make sure that:

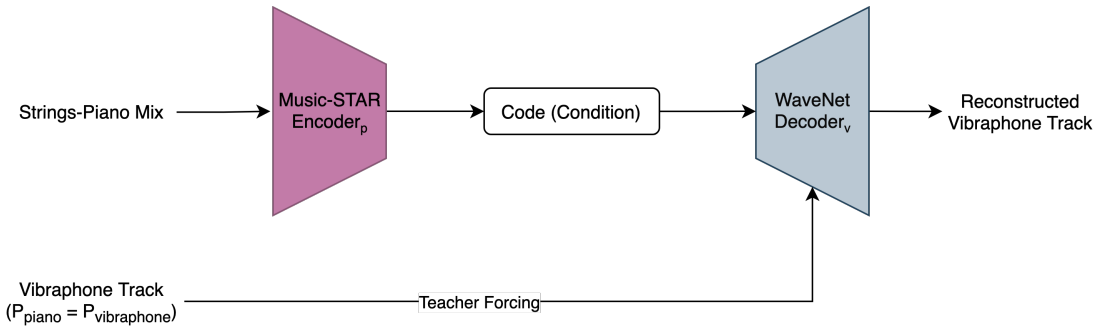

Figure 3.11: Training Music-STAR using the stem-supervised approach.

1. The encoder includes only P_{piano} in the code, removing the other three components of A_{mix} (T_{piano}, $T_{strings}$, and $P_{strings}$), which has been proven possible when removing the distortion in [75].

2. The decoder applies the vibraphone timbre to the output that depends on the audio segments we use in the teacher forcing procedure.

Accordingly, we conclude that if we:

1. Use the strings-piano mixture as the encoder's input;

2. Use the vibraphone counterpart of the piano track to train the decoder;

the autoencoder will finally be able to extract the piano component of any mixture and translate it into vibraphone (Figure 3.11).

The feasibility of this method depends on the existence of the paired data in our training set. When the decoder is learning to upsample the encoder's code to generate the vibraphone's segment, it inevitably guides the encoder to hand

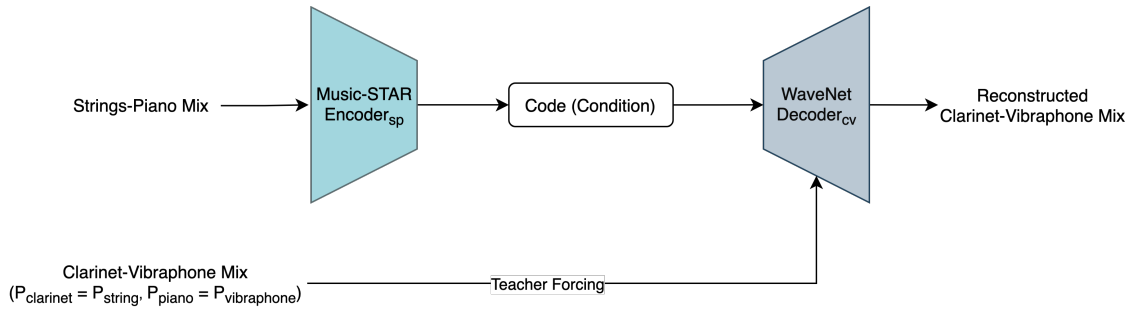

Figure 3.12: Training Music-STAR using the mixture-supervised approach.

in helpful information for the task, which should be the piano pitch information since it's the only feature the input and the target output have in common. This approach will apply to other instruments if included in the training set.

We consider two configurations in the supervised approach:

1. Stem-supervised, where the model takes a mixture as the input and outputs the translated version of one of the stems, as demonstrated in the example above. In this case, we employ one autoencoder for each of the instruments. (Figure 3.11)

2. Mixture-supervised, where the model takes a mixture as the input and outputs the translated mixture using one autoencoder. (Figure 3.12)

We examine the autoencoder's ability to detect the mutual features and remove unwanted information at different levels through these configurations.

By introducing stem-supervised and mixture-supervised Music-STAR, we take the applications of the WaveNet autoencoder even one step further to perform

audio-based music rearrangement.

3.4.2.1 Model Architecture

We use the famous WaveNet autoencoder architecture in the supervised approach where we replace the encoder with the Music-STAR encoder (Figure 3.9). The decoders' architecture is identical to the original WaveNet discussed in Section 3.2.

In stem-supervised Music-STAR, we employ two autoencoders, each for isolating and translating one of the instruments in the mixture, and combine their outputs in the end to obtain the rearranged mixture.

In mixture-supervised Music-STAR, the rearrangement takes place using one encoder and decoder in an end-to-end manner.

3.4.2.2 Model Training

We trained the models on the reduced version of StarNet quantized by 8-bit mu-law. We picked random one-second audio segments of a mixture file as the input and extract the same segment in the target tracks to use them for teacher forcing the decoders. The encoders and the decoders are trained jointly in stem-supervised and mixture-supervised models.

Stem-supervised loss function is formulated as

$$\sum_{j} L(D(E(mix_j), t_j)) \tag{3.8}$$

where L is the cross-entropy loss, D is the decoder, E represents the encoder, mix_j is the jth input sample, and t_j^i is the jth target sample performed by instrument i.

Mixture-supervised loss function is formulated as

$$\sum_{j} L(D(E(mix_j), t_j)) \tag{3.9}$$

where L is the cross-entropy loss, D is the decoder, E represents the encoder, mix_j is the jth input sample, and t_j is the jth sample from the target mixture.

Training is done using the Adam optimizer and exponential learning decay, where a learning rate of 3e$-$4 and a decay rate of 0.99 are applied. The model is trained for a total of 100 epochs with a batch size of 16.

The resulting rearranged piece during inference is formulated as below for the stem-supervised model:

$$\sum_{i \in I} D^i(E(mix)) \tag{3.10}$$

The mixture-supervised models is inferred as:

$$D(E(mix)) \tag{3.11}$$

4 Results and Evaluation

The previous chapter proposed Music-STAR, a system designed to perform multi-instrument music rearrangement, with three approaches: unsupervised, stem-supervised, and mixture-supervised. Music-STAR is able to translate music pieces from (into) two domains: strings-piano and clarinet-vibraphone. We also provided details on the design and implementation of our baseline models, single-instrument translation, and separation-translation pipeline. All the models mentioned above are based on WaveNet autoencoders.

Single-instrument translation is performed through finetuning the pre-trained models discussed in [75] with the StarNet dataset. The separation-translation pipeline leverages the Demucs network for the separation process and the same single-instrument translation models as above to generate the outputs. Unsupervised Music-STAR uses a Music-STAR encoder to extract the pitch information of one instrument track from a mixture and then employs the same decoders as in single-instrument translation to generate the audio. All three models mentioned

above use the same pre-trained WaveNet decoders for audio generation.

Stem-supervised Music-STAR performs a two-to-one translation where one of the stems in the input mixture is directly translated to its corresponding target instrument. Mixture-supervised Music-STAR takes a mixture as the input and generates the target mixture in an end-to-end manner.

In this chapter, we present the generated arrangements using these five models and evaluate them on three criteria:

1. Content preservation: how much of the pitch content of the input is retained in the output.

2. Style fit: How well the output presents the target timbres.

3. Audio quality: How clean and distortion-free the generated audio is.

We take two approaches in evaluating the results, i.e., subjective and objective. Subjective evaluation involves human participants who are asked to rank the outputs based on the criteria mentioned above. Objective evaluation adopts quantitative measures to compare the models' performances based on the same criteria.

The following sections provide the results, the details of the evaluation, and the comparison outcomes.

4.1 Results

We test and evaluate the models on ten music pieces, each presented as a 10-second audio file. Some of these pieces are from the StarNet dataset performed by both domains along with their stems. The rest are 10-second segments of famous music pieces obtained by applying the StarNet Instruments to their MIDI files. The audio samples are provided on the supplementary web page.

4.1.1 Single-Instrument Translation

The outputs of the single-instrument translation can be found here. In the case of clarinet-vibraphone to strings-piano translation, every clarinet track is translated into string ensemble, and every vibraphone track is translated into the corresponding piano version. The resulting tracks are mixed, forming the new arrangement. In the case of strings-piano to clarinet-vibraphone, the translation takes place contrariwise.

4.1.2 Separation-Translation Pipeline

The outputs from the separation-translation pipeline are presented here. In the case of clarinet-vibraphone to strings-piano translation, the input mixture is fed into the Demucs network. The isolated clarinet and vibraphone tracks are extracted and

then translated into strings and piano, respectively. By mixing the translation outputs, the new arrangement is delivered. In the case of strings-piano to clarinet-vibraphone, the translation takes place contrariwise.

4.1.3 Unsupervised Music-STAR

The outputs of unsupervised Music-STAR are provided here. In the case of clarinet-vibraphone to strings-piano translation, the input mixture is fed into two encoders, one of which extracts the pitch information of clarinet and the other extracts the pitch information of vibraphone. Then the piano and strings audio is generated by corresponding decoders and mixed together to form the new arrangement. A similar process applies to strings-piano to clarinet-vibraphone translation.

4.1.4 Stem-supervised Music-STAR

The outputs of stem-supervised Music-STAR are found here. In the case of clarinet-vibraphone to strings-piano translation, two autoencoders take in the input mixture, one outputs the piano track, while the other outputs the strings track. The two outputs are then mixed. Strings-piano to clarinet-vibraphone translation is possible in a similar manner.

4.1.5 Mixture-supervised Music-STAR

This link provides the outputs from mixture-supervised Music-STAR, in which the input mixture is directly translated into the target mixture using one autoencoder.

4.2 Evaluation

4.2.1 Subjective Evaluation

We carried out the subjective evaluation by distributing a survey (Appendix A) among 30 participants, some of whom had a musical background. Each survey contains one target mixture from each domain (strings-piano and clarinet-vibraphone) to demonstrate the gold standard, followed by corresponding translation outputs resulting from the five models. Since we have 10 test samples, each translated into two domains, ten surveys were created to cover all combinations. According to the number of participants, each target piece is assessed three times. The audio samples' order was different for the two pieces, and the participants had no prior knowledge of the order.

Every music piece in the survey is evaluated through four questions asking the participants to rank the five outputs (from (a) to (e)) based on:

1. How well they preserve the target musical content, which accounts for content preservation.

		String-Piano					Clarinet-Vibraphone				
	ranking	Single-Instrument	Separation-Translation	Unsupervised	Stem-supervised	Mixture-supervised	Single Instrument	Separation Translation	Unsupervised	Stem-supervised	Mixture-supervised
Content Preservation	1	0	2	2	8	18	7	6	3	0	14
	2	3	1	4	14	8	6	6	6	1	11
	3	7	11	8	3	1	9	6	10	4	1
	4	12	10	6	0	2	7	8	9	3	3
	5	8	6	10	5	1	1	4	2	22	1
Style Fit	1	2	1	2	8	17	3	1	4	2	20
	2	1	3	2	14	10	6	7	4	10	3
	3	8	7	11	4	0	10	6	11	1	2
	4	7	14	5	1	3	6	9	6	7	2
	5	12	5	10	3	0	5	7	5	10	3
Audio Quality	1	1	2	4	7	16	2	5	2	4	17
	2	1	2	1	15	11	7	8	5	3	7
	3	12	4	8	5	1	12	5	9	2	2
	4	4	18	4	2	2	5	5	9	9	2
	5	12	4	13	1	0	4	7	5	12	2
Overall	1	0	1	2	8	19	6	4	5	0	15
	2	1	0	4	15	10	9	9	4	2	6
	3	7	12	7	3	1	7	6	8	5	4
	4	10	11	6	3	0	4	6	9	8	3
	5	12	6	11	1	0	4	5	4	15	2

Figure 4.1: The models' rankings based on subjective evaluation

2. How well they present the instruments' sound colors (timbre) of the target piece, corresponding to style fit.

3. How clean and noise-free they are, presenting the audio quality.

4. Their overall quality compared to the target piece.

After collecting the surveys, we concluded the rankings by scoring the five models. Every model receives a score of 5 if one of its generated outputs is ranked first, a score of 4 if ranked second, a score of 3 if ranked third, a score of 2 if ranked fourth, and a score of 1 if ranked last.

Figure 4.1 demonstrates the number of times each model gets a specific ranking in terms of content preservation, style fit, audio quality, and overall quality for

both domains. For instance, out of thirty participants, eighteen people ranked mixture-supervised Music-STAR first in content preservation for the strings-piano domain. Eight participants ranked it second, while only one person found it to be the third. Two people picked it as the fourth model, and one person ranked it as the last model. Accordingly, the score of mixture-supervised Music-STAR in terms of content preservation will be: $18 * 5 + 8 * 4 + 1 * 3 + 2 * 2 + 1 * 1 = 130$

Figure 4.2 shows the scores of all five models for both domains separately and in total regarding the four questions in the survey. According to the final scores, mixture-supervised Music-STAR is performing best in all four criteria for both domains. While stem-supervised Music-STAR is the second-best model for the strings-piano domain, it cannot compete with other models in clarinet-vibraphone translation. On the other hand, single-instrument translation has been the second-best model for clarinet-vibraphone except for audio quality measure as it is placed after the separation-translation pipeline.

4.2.2 Objective Evaluation

The subjective evaluation provides a qualitative assessment of our models mainly based on how users perceive the generated outputs. We aim to provide a quantitative quality assessment using techniques employed in previous studies through the objective evaluation. To this end, we address the same criteria, i.e., content

	Model	String-Piano	Clarinet-Vibraphone	Total
Content Preservation	Single-Instrument	65	101	166
	Separation-Translation	73	92	165
	Unsupervised Music-STAR	72	89	161
	Stem-supervised Music-STAR	110	44	154
	Mixture-supervised Music-STAR	130	124	254
Style Fit	Single-Instrument	64	86	150
	Separation-Translation	71	76	147
	Unsupervised Music-STAR	71	86	157
	Stem-supervised Music-STAR	113	77	190
	Mixture-supervised Music-STAR	131	125	256
Audio Quality	Single-Instrument	65	88	153
	Separation-Translation	70	89	159
	Unsupervised Music-STAR	69	80	149
	Stem-supervised Music-STAR	115	68	183
	Mixture-supervised Music-STAR	131	125	256
Overall	Single-Instrument	57	99	156
	Separation-Translation	69	91	160
	Unsupervised Music-STAR	70	87	157
	Stem-supervised Music-STAR	116	54	170
	Mixture-supervised Music-STAR	138	119	257

Figure 4.2: The models' scores based on subjective evaluation

preservation, style fit, and audio quality, the details of which are discussed below.

4.2.2.1 Content Preservation

Following the work by Cífka et al. [15], we assess the models on content preservation by calculating the Jaccard similarity between the pitch contours of the outputs and their corresponding gold standards. Pitch contours provide the perceived pitches in a sound signal over time. Since we are addressing multi-instrument music in our study, we extract the pitch contours using the multi-pitch Melodia algorithm [90] which provides the existing pitch frequencies in hertz. We round the frequency values to the nearest semitone. Then we express the similarity of the pitch sets of each time step (A and B) in terms of the Jaccard index:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.1}$$

Higher Jaccard similarity between the output and the gold standard signifies better content preservation. Figure 4.3 reports the average Jaccard similarity of the five models for both domains separately and in total. According to this measure, mixture-supervised Music-STAR surpasses the other models in terms of content preservation. As shown in the figure, the other models perform differently in the case of the two domains.

Model	String-Piano	Clarinet-Vibraphone	Average
Single-Instrument	0.372	0.370	0.371
Separation-Translation	0.406	0.378	0.392
Unsupervised Music-STAR	0.348	0.351	0.350
Stem-supervised Music-STAR	0.424	0.222	0.323
Mixture-supervised Music-STAR	0.463	0.389	0.426

Figure 4.3: Jaccard similarity between pitch contours for content preservation assessment.

4.2.2.2 Style Fit

We evaluate the style fit factor using the deep metric triplet network offered by Lee et al. [67]. The network consists of a backbone model, which provides an embedding for the three inputs, and a triplet model that outputs a similarity score between the three embeddings. The three inputs are called the anchor, the positive, and the negative input. The goal is to output a similarity score between the anchor and the other inputs. It is expected that the anchor resembles the positive input while being considerably different from the negative input.

Following Cífka et al. [15], we use MFCCs as the input features. As mentioned before, MFCCs provide timbre-related information of the input audio. We trained the triplet network using the mixtures in the StarNet dataset. The MFCCs of eight-second clarinet-vibraphone audio segments were used as both the anchors and the

Model	String-Piano	Clarinet-Vibraphone	Average
Single-Instrument	0.504	0.462	0.483
Separation-Translation	0.469	0.479	0.474
Unsupervised Music-STAR	0.495	0.448	0.472
Stem-supervised Music-STAR	0.783	0.615	0.699
Mixture-supervised Music-STAR	0.773	0.623	0.698

Figure 4.4: The cosine similarities between MFCCs for style fit assessment

positive inputs in the training phase. At the same time, their counterparts from the strings-piano domain were considered as the negative inputs.

During inference, we presented the translation outputs from the five models as the anchors, their corresponding gold standard as the positive input, and the performance from the other domain as the negative input.

Figure 4.4 reports the average cosine similarity between the outputs of each model and their corresponding gold standards. Higher cosine similarity denotes more likeness to the target timbre and a better style fit. Stem-supervised and mixture-supervised Music-STAR have comparable results in both domains, performing as the best two models in terms of style fit. The other three models perform differently for the two domains.

4.2.2.3 Audio Quality

We take advantage of the source-to-distortion ratio (SDR) [104] to evaluate the audio quality of the translation outputs. SDR metric has been widely used to assess the quality of audio source separation models. It provides an overall measure by involving error terms for interference (e_{interf}), noise (e_{noise}), and additive artifacts (e_{artif}) which are distorting the target signal (s_{target}):

$$\hat{s}_j = s_{target} + e_{interf} + e_{noise} + e_{artif} \tag{4.2}$$

In our work, interference accounts for the amount of the source instruments that can be heard in the output, noise is caused by the sensors such as the microphones, and artifacts are the sonic material created due to audio manipulation.

SDR is reported in decibels and formulated as :

$$SDR = 10\log_{10}(\frac{\|s_{target}\|^2}{\|e_{interf} + e_{noise} + e_{artif}\|^2}) \tag{4.3}$$

We calculate the SDR of every generated output compared to its corresponding gold standard. The higher the SDR value, the better is the sound quality. As shown in Figure 4.5, mixture-supervised Music-STAR has the highest SDR for both domains. The other models perform differently for the two domains.

Model	String-Piano	Clarinet-Vibraphone	Average
Single-Instrument	-17.34	-9.89	-13.62
Separation-Translation	-15.28	-10.55	-12.91
Unsupervised Music-STAR	-16.43	-12.72	-14.57
Stem-supervised Music-STAR	-9.95	-12.24	-11.09
Mixture-supervised Music-STAR	-8.37	-7.39	-7.88

Figure 4.5: SDR metric for audio quality assessment

4.2.3 Discussion

So far, we have presented the results from both objective and subjective evaluations. In this section, we investigate if the two evaluation methods agree with each other. To this end, we compare the models' rankings achieved through the two evaluation methods.

4.2.3.1 Content Preservation

Figure 4.6 shows the final rankings of the five models on content preservation. The evaluation methods have a 60% agreement on the rankings. Their differences mainly lie in the performances of single-instrument translation, separation-translation pipeline, and unsupervised Music-STAR. As shown in Figure 4.3 the reported similarity rates of these three models are not considerably different. Since

	String-Piano rankings		Clarinet-Vibraphone rankings		Overall rankings	
Model	**Subjective**	**Objective**	**Subjective**	**Objective**	**Subjective**	**Objective**
Mixture-supervised Music-STAR	1	1	1	1	1	1
Stem-supervised Music-STAR	2	2	5	5	5	5
Unsupervised Music-STAR	4	5	4	4	4	4
Separation-Translation	3	3	3	2	3	2
Single-Instrument	5	4	2	3	2	3

Figure 4.6: The models' rankings on content preservation

the first two models use the same universal encoder to capture the pitch information and the encoder in unsupervised Music-STAR is trained based on the universal encoder, this analogy seems reasonable.

According to the final rankings, mixture-supervised Music-STAR puts in the best performance in terms of content preservation.

4.2.3.2 Style Fit

The models' rankings on style fit are shown in Figure 4.7. The two evaluation methods seem not to agree about any of the models' overall performance. However, there is an explanation for this matter. By referring to Figure 4.4, we can see that the difference between mixture-supervised and stem-supervised Music-STAR in their average performance is only 0.001 that can be definitely ignored. As a

	String-Piano rankings		Clarinet-Vibraphone rankings		Overall rankings	
Model	**Subjective**	**Objective**	**Subjective**	**Objective**	**Subjective**	**Objective**
Mixture-supervised Music-STAR	1	2	1	1	1	2
Stem-supervised Music-STAR	2	1	3	2	2	1
Unsupervised Music-STAR	3	4	2	5	3	5
Separation-Translation	3	5	4	3	5	4
Single-Instrument	4	3	2	4	4	3

Figure 4.7: The models' rankings on style fit

result, we can assume that the two models perform comparably in terms of style fit.

Moreover, we mentioned earlier that single-instrument translation, separation-translation pipeline, and unsupervised Music-STAR share the same WaveNet decoders and that the decoders are in charge of applying the target timbres. As a result, we can see in Figure 4.4 that the three models have a close contest with each other, and this can justify the mismatch between the evaluation results.

4.2.3.3 Audio Quality

The models are ranked based on their audio quality in Figure 4.8. The two evaluation methods agree 100% on the overall rankings, demonstrating the superiority of mixture-supervised Music-STAR.

	String-Piano rankings		Clarinet-Vibraphone rankings		Overall rankings	
Model	**Subjective**	**Objective**	**Subjective**	**Objective**	**Subjective**	**Objective**
Mixture-supervised Music-STAR	1	1	1	1	1	1
Stem-supervised Music-STAR	2	2	5	4	2	2
Unsupervised Music-STAR	4	4	4	5	5	5
Separation-Translation	3	3	2	3	3	3
Single-Instrument	5	5	3	2	4	4

Figure 4.8: The models' rankings on audio quality

4.2.3.4 Discussion Summary

We conclude the evaluation as follows:

1. Our evaluation outcomes conclude that mixture-supervised Music-STAR is the predominant model in performing multi-instrument music translation. This is most likely caused by training the model in a supervised manner and the fact that the encoders are not shared among the domains. Although training an unsupervised model is much easier on the dataset and employing a universal encoder gives us a more generalized model, it might all come at the cost of performance, and the results from our baseline models confirm this.

2. Although we expected stem-supervised Music-STAR to have a similar or even

better performance than mixture-supervised Music-STAR, it failed to execute the clarinet-vibraphone translation at high quality. We will investigate the reasons for this discrepancy as a part of our future work.

3. Unsupervised Music-STAR is, on average, the worst-performing model, and it is acceptable for one main reason. The Music-STAR encoder used in this model is trained based on the universal encoder's outputs to extract the pitch information of one instrument out of a mixture. Having an imperfect ground truth places the model at the disadvantage of even more unsatisfactory performance. Besides, unsupervised Music-STAR uses the same decoders as the baseline models, which brings it no additional advantage.

4. The rankings of single-instrument and separation-translation models on different criteria seem exchangeable as they display a very similar translation quality. Thanks to the great source separation quality by Demucs, the separation-translation pipeline could produce quite flawless inputs for the translation model.

5 Conclusion

This thesis aimed to investigate the feasibility of multi-instrument music translation establishing music rearrangement as the ultimate goal. To this end, we proposed Music-STAR, a system implemented through three approaches to conduct multi-instrument music translation. In this chapter, we summarize the contributions of our research and discuss how it can be expanded in the future.

5.1 Thesis Contributions

We enumerate the major contributions of this thesis as follows.

1. In this thesis, we introduced the only audio-based multi-instrument music translation system in which every stem is translated into a specified target instrument. Our goal was to tackle this problem without applying explicit source separation to the input mixtures. Accordingly, we proposed Music-STAR, a system with an autoencoder as the backbone that makes both source separation and pitch-timbre disentanglement possible in the latent space.

2. To address multi-instrument music translation, we proposed Music-STAR with three models: unsupervised, stem-supervised, and mixture-supervised. The three models are based on WaveNet autoencoders, where a Music-STAR encoder replaces the original encoder. Incorporating more layers and non-linearities makes the Music-STAR encoder powerful enough to process the mixture inputs.

3. In order to train and test Music-STAR models, we required a dataset that includes music pieces performed by different sets of instruments along with their stems. Therefore, we created StarNet, an audio dataset containing roughly nine hours of classical music pieces performed in two domains: strings-piano and clarinet-vibraphone.

4. We considered two baseline models to compare Music-STAR's outputs to. The first is applying single-instrument translation to the stems and then mixing the outputs, which is only applicable when the stems are available. The second is to separate the stems using music source separation models and then apply single-instrument translation, which we referred to as a separation-translation pipeline. To bring these models into play, we needed to a) finetune the existing single-instrument translation model on StarNet to support the exact timbres we account for in the dataset, b) to retrain the source separation model to

separate the stems of strings-piano and clarinet-vibraphone mixtures.

5. After training the baseline and Music-STAR models, we assessed the outputs through subjective and objective evaluations. Although the models performed differently on the two mixture domains, both evaluation methods agreed that mixture-supervised Music-STAR is the best approach toward multi-instrument music translation overall.

5.2 Future Work

Our work is the first attempt toward audio-based multi-instrument music translation, and there are many other questions and problems to be addressed in the future. For instance:

1. Increasing the number of instrument tracks in the mixtures.

2. Adding to the variety of instrument combinations in the dataset and including music genres other than classical music.

3. Accounting for voice-to-instrument translation or enable voice conversion to make the whole idea of cover song generation possible.

Bibliography

[1] Grigory Antipov, Moez Baccouche, and Jean-Luc Dugelay. Face aging with conditional generative adversarial networks. In *2017 IEEE international conference on image processing (ICIP)*, pages 2089–2093. IEEE, 2017.

[2] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *arXiv preprint arXiv:2003.05991*, 2020.

[3] Juan Pablo Bello, Giuliano Monti, Mark B Sandler, et al. Techniques for automatic music transcription. In *ISMIR*, 2000.

[4] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30, 2018.

[5] Adrien Bitton, Philippe Esling, and Axel Chemla-Romeu-Santos. Modulated variational auto-encoders for many-to-many musical timbre transfer. *arXiv preprint arXiv:1810.00222*, 2018.

[6] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep learning techniques for music generation–a survey. *arXiv preprint arXiv:1709.01620*, 2017.

[7] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neu-

ral photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

[8] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[9] Gino Brunner, Andres Konrad, Yuyi Wang, and Roger Wattenhofer. Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer. *arXiv preprint arXiv:1809.07600*, 2018.

[10] Gino Brunner, Nawel Naas, Sveinn Palsson, Oliver Richter, and Roger Wattenhofer. Monaural music source separation using a resnet latent separator network. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1124–1131. IEEE, 2019.

[11] Estefania Cano, Derry FitzGerald, Antoine Liutkus, Mark D Plumbley, and Fabian-Robert Stöter. Musical source separation: An introduction. *IEEE Signal Processing Magazine*, 36(1):31–40, 2018.

[12] Ali Taylan Cemgil, Hilbert J Kappen, and David Barber. A generative model for music transcription. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):679–694, 2006.

[13] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[14] François Chollet. Xception: Deep learning with depthwise separable convolu-

tions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[15] Ondřej Cífka, Alexey Ozerov, Umut Şimşekli, and Gaël Richard. Self-supervised vq-vae for one-shot music style transfer. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 96–100. IEEE, 2021.

[16] Ondřej Cífka, Umut Şimşekli, and Gaël Richard. Supervised symbolic music style translation using synthetic data. *arXiv preprint arXiv:1907.02265*, 2019.

[17] Ondřej Cífka, Umut Şimşekli, and Gaël Richard. Groove2groove: One-shot music style transfer with supervision from synthetic data. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2638–2650, 2020.

[18] David Cope. *The algorithmic composer*, volume 16. AR Editions, Inc., 2000.

[19] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[20] Shuqi Dai, Zheng Zhang, and Gus G Xia. Music style transfer: A position paper. *arXiv preprint arXiv:1803.06841*, 2018.

[21] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR, 2017.

[22] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. Music source separation in the waveform domain. *arXiv preprint arXiv:1911.13254*, 2019.

[23] Alexandre Défossez, Neil Zeghidour, Nicolas Usunier, Léon Bottou, and Francis Bach. Sing: Symbol-to-instrument neural generator. *arXiv preprint arXiv:1810.09785*, 2018.

[24] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[25] Hui Ding, Kumar Sricharan, and Rama Chellappa. Exprgan: Facial expression editing with controllable expression intensity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[26] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*, 2018.

[27] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[28] Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms. *arXiv preprint arXiv:1706.07068*, 2017.

[29] Daniel PW Ellis. Extracting information from music audio. *Communications of the ACM*, 49(8):32–37, 2006.

[30] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*, 2019.

[31] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musi-

cal notes with wavenet autoencoders. In *International Conference on Machine Learning*, pages 1068–1077. PMLR, 2017.

[32] Slim Essid, Gaël Richard, and Bertrand David. Musical instrument recognition by pairwise classification strategies. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1401–1412, 2006.

[33] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[34] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[35] Leon A Gatys, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Preserving color in neural artistic style transfer. *arXiv preprint arXiv:1606.05897*, 2016.

[36] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[37] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[38] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.

[39] Zhiyuan Guo, Qiang Wang, Gang Liu, and Jun Guo. A query by humming system based on locality sensitive hashing indexes. *Signal Processing*, 93(8):2229–2243, 2013.

[40] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. Deepbach: a steerable model for bach chorales generation. In *International Conference on Machine Learning*, pages 1362–1371. PMLR, 2017.

[41] Yoonchang Han, Jaehun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):208–221, 2016.

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[43] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[44] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[45] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[46] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[47] Yedid Hoshen, Ron J Weiss, and Kevin W Wilson. Speech acoustic modeling from raw multichannel waveforms. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4624–4628. IEEE, 2015.

[48] Sicong Huang, Qiyang Li, Cem Anil, Xuchan Bao, Sageev Oore, and Roger B Grosse. Timbretron: A wavenet (cyclegan (cqt (audio))) pipeline for musical timbre transfer. *arXiv preprint arXiv:1811.09620*, 2018.

[49] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 172–189, 2018.

[50] Yun-Ning Hung, I Chiang, Yi-An Chen, Yi-Hsuan Yang, et al. Musical composition style transfer via disentangled timbre representations. *arXiv preprint arXiv:1905.13567*, 2019.

[51] Muhammad Huzaifah and Lonce Wyse. Deep generative models for musical audio synthesis. *arXiv preprint arXiv:2006.06426*, 2020.

[52] Daniel Jiwoong Im, Chris Dongjoo Kim, Hui Jiang, and Roland Memisevic. Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110*, 2016.

[53] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[54] Deepak Kumar Jain, Akshi Kumar, Linqin Cai, Siddharth Singhal, and Vaib-

hav Kumar. Att: Attention-based timbre transfer. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2020.

[55] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*, 2018.

[56] Takuhiro Kaneko and Hirokazu Kameoka. Cyclegan-vc: Non-parallel voice conversion using cycle-consistent adversarial networks. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 2100–2104. IEEE, 2018.

[57] Takuhiro Kaneko, Hirokazu Kameoka, Kou Tanaka, and Nobukatsu Hojo. Cyclegan-vc2: Improved cyclegan-based non-parallel voice conversion. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6820–6824. IEEE, 2019.

[58] Takuhiro Kaneko, Hirokazu Kameoka, Kou Tanaka, and Nobukatsu Hojo. Cyclegan-vc3: Examining and improving cyclegan-vcs for mel-spectrogram conversion. *arXiv preprint arXiv:2010.11672*, 2020.

[59] Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954, 1960.

[60] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[61] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[62] Oleksii Kuchaiev and Boris Ginsburg. Training deep autoencoders for collaborative filtering. *arXiv preprint arXiv:1708.01715*, 2017.

[63] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. *arXiv preprint arXiv:1910.06711*, 2019.

[64] Cyril Laurier, Jens Grivolla, and Perfecto Herrera. Multimodal music mood classification using audio and lyrics. In *2008 Seventh International Conference on Machine Learning and Applications*, pages 688–693. IEEE, 2008.

[65] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[66] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[67] Jongpil Lee, Nicholas J Bryan, Justin Salamon, Zeyu Jin, and Juhan Nam. Disentangled multidimensional metric learning for music similarity. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6–10. IEEE, 2020.

[68] Alexander Lerch. *An introduction to audio content analysis: Applications in signal processing and music informatics*. Wiley-IEEE Press, 2012.

[69] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *arXiv preprint arXiv:1703.00848*, 2017.

[70] Chien-Yu Lu, Min-Xin Xue, Chia-Che Chang, Che-Rung Lee, and Li Su. Play as you like: Timbre-enhanced multi-modal music style transfer. In *Proceedings*

of the AAAI Conference on Artificial Intelligence, volume 33, pages 1061–1068, 2019.

[71] Wei Tsung Lu, Li Su, et al. Transferring the style of homophonic music using recurrent neural networks and autoregressive model. In *ISMIR*, pages 740–746, 2018.

[72] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[73] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.

[74] James A Moorer. On the transcription of musical sound by computer. *Computer Music Journal*, pages 32–38, 1977.

[75] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A universal music translation network. *arXiv preprint arXiv:1805.07848*, 2018.

[76] Meinard Müller. *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.

[77] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[78] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.

[79] Dimitri Palaz, Ronan Collobert, and Mathew Magimai Doss. Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks. *arXiv preprint arXiv:1304.1018*, 2013.

[80] Yingxue Pang, Jianxin Lin, Tao Qin, and Zhibo Chen. Image-to-image translation: Methods and applications. *arXiv preprint arXiv:2101.08629*, 2021.

[81] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[82] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The MUSDB18 corpus for music separation, December 2017.

[83] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, pages 1060–1069. PMLR, 2016.

[84] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[85] Robert Rowe. *Machine musicianship*. MIT press, 2004.

[86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[87] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[88] Matti Ryynanen and Anssi Klapuri. Query by humming of midi and audio using locality sensitive hashing. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2249–2252. IEEE, 2008.

[89] Tara Sainath, Ron J Weiss, Kevin Wilson, Andrew W Senior, and Oriol Vinyals. Learning the speech front-end with raw waveform cldnns. 2015.

[90] Justin Salamon and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1759–1770, 2012.

[91] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[92] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*, pages 111–112, 2015.

[93] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[94] Berrak Sisman, Junichi Yamagishi, Simon King, and Haizhou Li. An overview of voice conversion and its challenges: From statistical modeling to deep learn-

ing. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2020.

[95] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[96] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. *arXiv preprint arXiv:1806.03185*, 2018.

[97] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji. Open-unmix-a reference implementation for music source separation. *Journal of Open Source Software*, 4(41):1667, 2019.

[98] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

[99] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[100] John Thickstun, Zaid Harchaoui, and Sham M. Kakade. Learning features of music from scratch. In *International Conference on Learning Representations (ICLR)*, 2017.

[101] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

[102] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756. PMLR, 2016.

[103] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[104] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE transactions on audio, speech, and language processing*, 14(4):1462–1469, 2006.

[105] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[106] Junhao Wang, Cong Jin, Wei Zhao, Shan Liu, and Xin Lv. An unsupervised methodology for musical style translation. In *2019 15th International Conference on Computational Intelligence and Security (CIS)*, pages 216–220. IEEE, 2019.

[107] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European conference on computer vision*, pages 597–613. Springer, 2016.

[108] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

A User Study Survey

The following pages include an examples of the survey used for subjective evaluation.

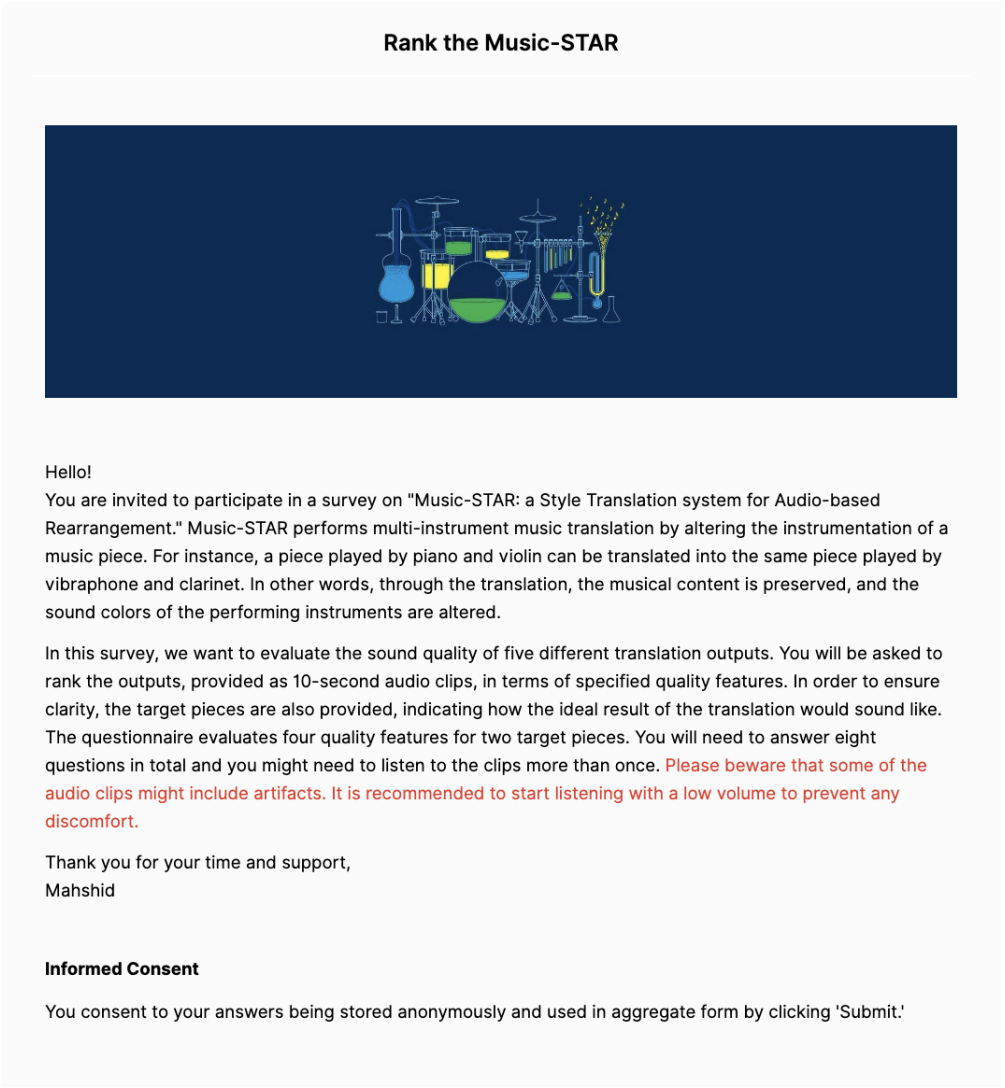

Figure A.1: Subjective evaluation survey - description and consent

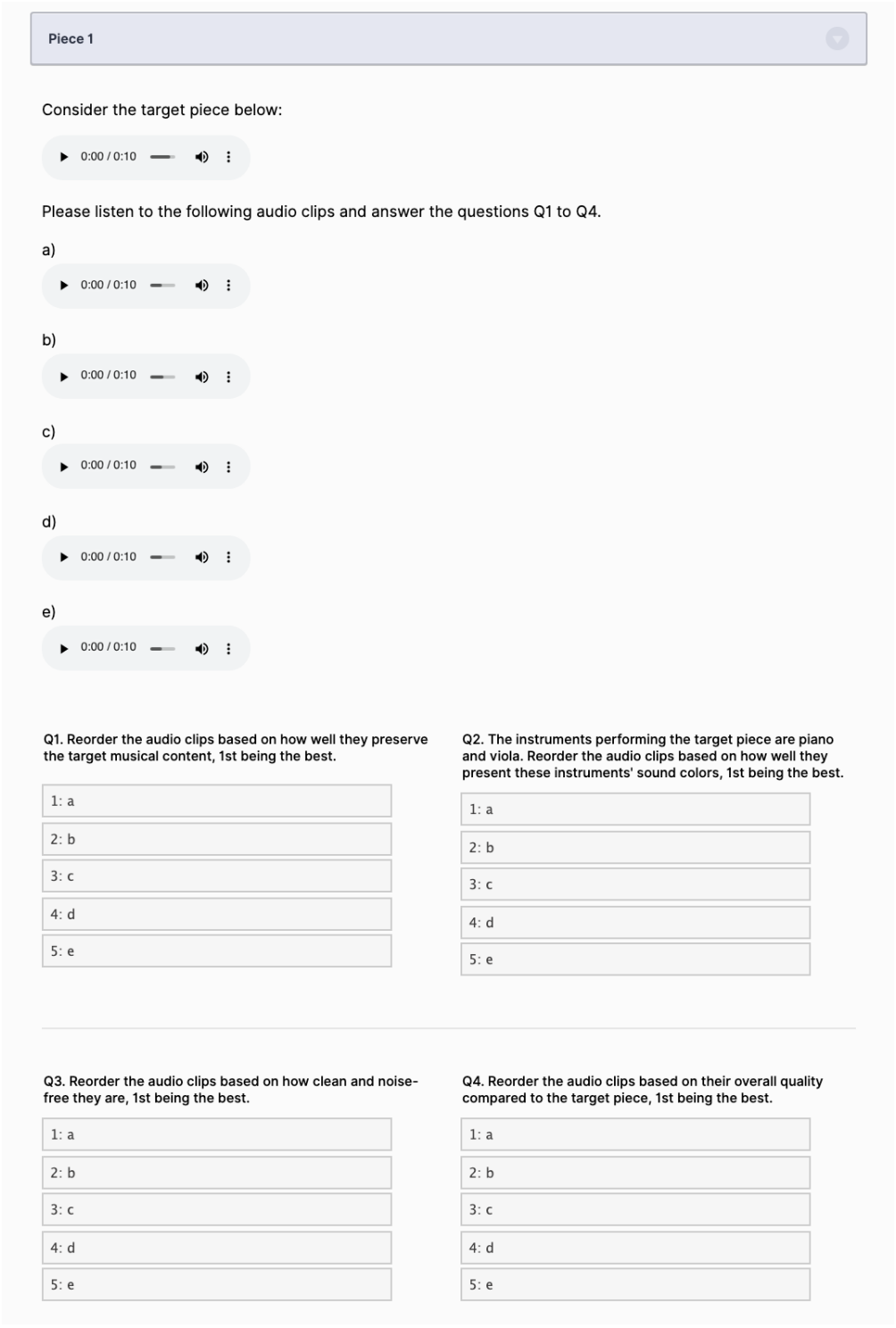

Figure A.2: Subjective evaluation survey - piece 1 questions

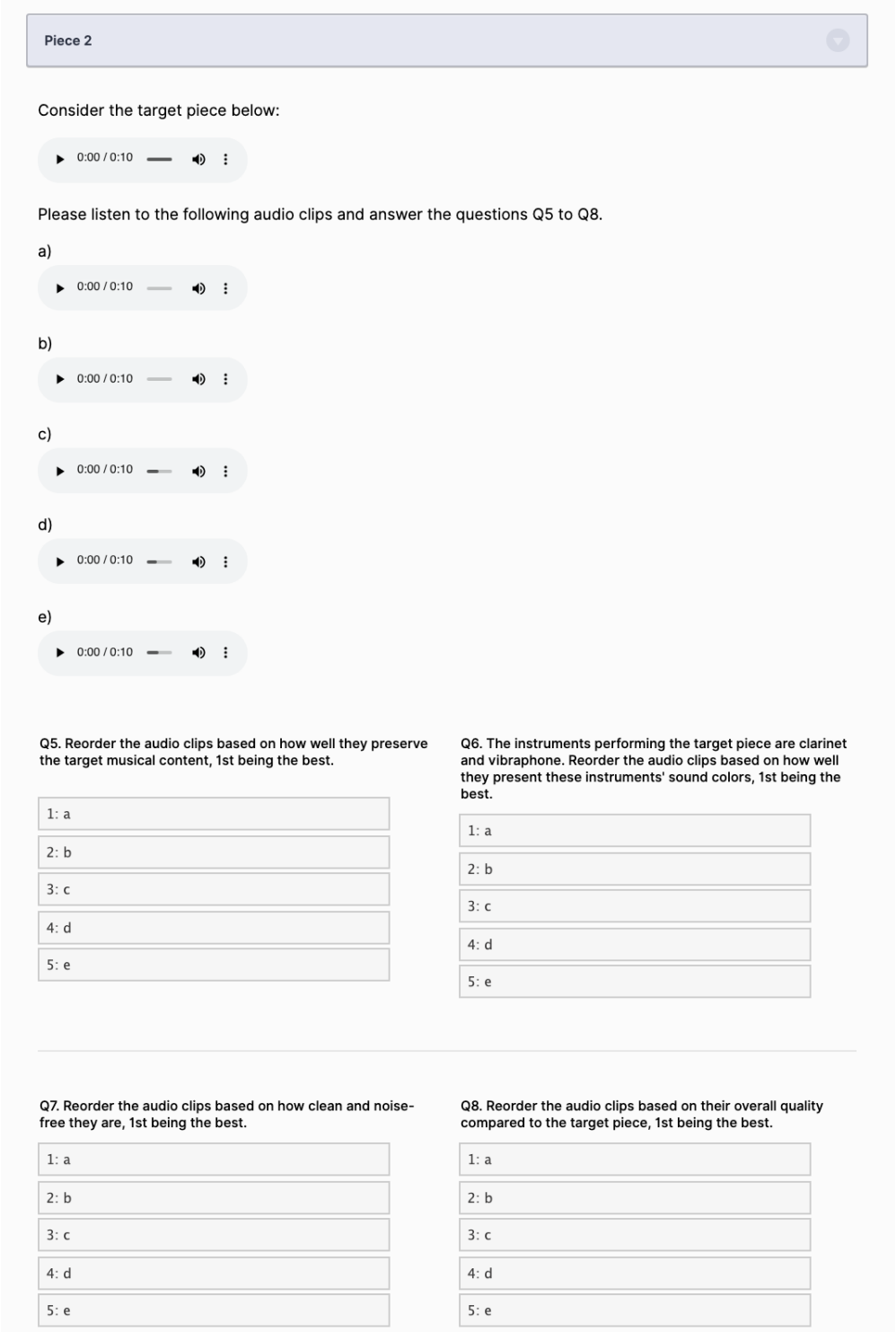

Figure A.3: Subjective evaluation survey - piece 2 questions