MASTER'S THESIS 2026:E18

Latent Space Interpolation for Music Transitions

Olle Rydén

ISSN: 1404-6342
LUTFMS-3551-2026

MATHEMATICAL STATISTICS
CENTRE FOR MATHEMATICAL SCIENCES
LTH | LUND UNIVERSITY

Latent Space Interpolation for Music Transitions

Olle Rydén

Master's thesis
2026:E18

Lund Institute of Technology
Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

Centrum Scientiarum Mathematicarum

Master's Theses in Mathematical Sciences 2026:E18
ISSN 1404-6342

LUTFMS-3551-2026

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/

Latent Space Interpolation for Music Transitions

Interpolering i det Latenta Rummet för Musikövergångar

Olle Rydén
ol1856ry-s@student.lu.se; ollerydenn@gmail.com

April 14, 2026

Master's thesis work carried out at the
Centre for Mathematical Sciences, Mathematical Statistics
Lund University.

Supervisor: Ted Kronvall, ted.kronvall@matstat.lu.se

Examiner: Maria Sandsten, maria.sandsten@matstat.lu.se

Abstract

Music generation has progressed rapidly in recent years, but practical track-to-track transitions are still largely handled with conventional signal processing such as equal-power crossfades and manual beatmatching. This thesis evaluates whether latent-space interpolation in a pretrained neural audio codec can serve as a short, targeted generation operator for music transitions, where the goal is a coherent bridge rather than a simple overlap of two songs. Using EnCodec, bar-aligned segments are tempo-aligned (with optional pitch preservation), encoded to a latent timeline, and blended frame-wise using different interpolation schedules (linear, sigmoid/eased, and spherical linear interpolation). The resulting decoded transitions are compared against baseline equal-power waveform crossfades built on the same alignment pipeline.

Evaluation combines objective similarity metrics, latent-space diagnostics, and listening tests. Time–frequency coherence (with cross-correlation alignment) is used to compare latent transitions to the baseline, while Principal Component Analysis trajectories, correlation structure, and per-channel frequency contribution probes are used to characterize how EnCodec represents musical structure during interpolation. In an A/B listening test (AB) across multiple transitions ($N = 14$), latent interpolation (spherical interpolation; SLERP) was preferred at a similar rate as the baseline, with a substantial fraction of "no preference" responses, indicating comparable perceptual quality rather than a clear winner. Diagnostics further suggest structured latent trajectories and systematic channel behavior, and indicate that latent magnitude correlates with decoded power with a small temporal offset. Overall, latent interpolation in a neural codec provides a viable alternative transition operator and motivates future work on learned refinement, automatic transition point selection, and conditional generative inpainting over codec latents.

Keywords: Neural Audio Codecs, EnCodec, Latent Space Interpolation, Music Transitions, Cross-correlation Analysis, Time-Frequency Coherence, DJ Transition Automation, Generative Music

Contents

1 Introduction **7**
- 1.1 Objective . . . 8
- 1.2 Contribution . . . 8
- 1.3 Related Work . . . 8
  - 1.3.1 Music Models over Audio Tokens . . . 8
  - 1.3.2 Autoencoders, Codecs, and Latent Manipulation . . . 9
  - 1.3.3 Diffusion and Inpainting for Audio Editing . . . 9
  - 1.3.4 Applications in Music Transitions and Blending . . . 10

2 Theoretical Background **11**
- 2.1 Digital Audio Representation . . . 11
  - 2.1.1 Sampling and Quantization . . . 11
  - 2.1.2 Frame-Based Processing and Windowing . . . 12
  - 2.1.3 Time-Frequency Analysis . . . 12
  - 2.1.4 Reconstruction Artifacts . . . 13
- 2.2 Music Theory Background . . . 13
  - 2.2.1 Musical Structure . . . 13
  - 2.2.2 Tempo, Beats and Bars . . . 14
  - 2.2.3 Key, Tonality and Harmonic Compatibility . . . 14
- 2.3 Conventional Transition Techniques . . . 15
  - 2.3.1 Beatmatching and Tempo Alignment . . . 15
  - 2.3.2 Time-Stretching Methods . . . 15
  - 2.3.3 Crossfades and Equal-Power Blending . . . 15
  - 2.3.4 Adhering to Musical Structure . . . 16
- 2.4 Neural Networks and Backpropagation . . . 16
  - 2.4.1 Neural Network Building Blocks . . . 16
  - 2.4.2 Convolutional Layers . . . 17
  - 2.4.3 Recurrent Layers (LSTM) . . . 17
- 2.5 Neural Audio Codecs and Latent Spaces . . . 17
  - 2.5.1 Autoencoders . . . 18

2.5.2 Vector Quantization . . . 18
2.5.3 EnCodec . . . 19
2.6 Latent Interpolation Methods . . . 23
2.6.1 Linear Interpolation . . . 24
2.6.2 Spherical Linear Interpolation (SLERP) . . . 24
2.6.3 Sigmoid/Eased Interpolation Curve . . . 24
2.6.4 Latent Alignment and Scale Handling . . . 25

3 Method **27**
3.1 Data and Shared Preparation . . . 27
3.1.1 Tracks and selection criteria . . . 27
3.1.2 Shared preprocessing and transition preparation . . . 27
3.2 Transition Pipelines . . . 28
3.2.1 Baseline transition . . . 28
3.2.2 EnCodec Latent Interpolation Pipeline . . . 29
3.3 Transition-quality evaluation . . . 31
3.3.1 Model selection for further analysis . . . 31
3.3.2 Listening tests . . . 32
3.3.3 Objective similarity metrics . . . 32
3.3.4 Qualitative audio diagnostics . . . 33
3.4 Latent and diagnostic analyses . . . 33
3.4.1 Latent trajectory diagnostics . . . 33
3.4.2 Latent structure diagnostics . . . 34
3.4.3 Pitch-coding diagnostics . . . 35
3.4.4 Latent magnitude vs decoded power . . . 36
3.5 Implementation . . . 37
3.5.1 Exact setups . . . 37
3.5.2 Software Stack, Hardware . . . 37
3.5.3 Inference Settings and Runtime . . . 37

4 Results **39**
4.1 Transition-quality evaluation . . . 39
4.1.1 Listening tests . . . 39
4.1.2 Objective similarity metrics . . . 41
4.1.3 Qualitative audio diagnostics . . . 44
4.2 Latent and diagnostic analyses . . . 46
4.2.1 Latent trajectory diagnostics . . . 46
4.2.2 Latent structure diagnostics . . . 49
4.2.3 Pitch-coding diagnostics . . . 50
4.2.4 Latent magnitude vs decoded power . . . 54

5 Discussion **57**
5.1 Summary of Key Results . . . 57
5.2 Interpreting Transition Quality vs. the Baseline . . . 58
5.3 What the Diagnostics Suggest About EnCodec Latents . . . 60
5.4 Practical Considerations and Limitations . . . 62

6 Conclusions **63**
6.1 Main Conclusions 63
6.2 Practical Implications 63
6.3 Future Work 64

References **67**

Appendix A Abbreviations **73**

Appendix B Supplementary Neural-Network Background **75**

Appendix C Declaration of AI usage in this thesis **77**

Appendix D Repository **79**

Chapter 1
Introduction

Smooth transitions between two recorded tracks are a practical and recurring objective in disc jockey (DJ) performance, radio automation, and playlist sequencing. In these settings, the goal is not to compose new music from scratch, but to move from an outgoing song to an incoming song without disrupting rhythm, phrasing, or perceived energy. Most automated systems therefore rely on classical signal processing: beat/bar alignment and gain-based mixing such as equal-power crossfades, along with equalizer effects. These tools are robust and predictable, but they are also limited to operations that are explicitly defined in the waveform domain, and they do not offer a principled way to *transform* one mixture into another.

This thesis explores an alternative viewpoint, which is to treat a pretrained neural audio codec as a compact working representation, and perform the transition *inside* that compressed space. This space is what we refer to as the *latent space*. Modern neural codecs map waveform audio to a downsampled latent timeline that retains perceptually important structure while discarding redundancy. This latent sequence can be interpreted as a time-indexed set of feature vectors that the decoder can re-synthesize back into audio. If two tracks are first aligned to a shared musical grid (bars/beats), the corresponding latent frames can be interpolated frame-wise over a chosen fade window and decoded into a single, continuous waveform. In this framing, interpolation is not "generating a new song", but instead a transformation that uses a learned, perceptually shaped representation of audio to blend two fixed endpoints.

A second motivation is scientific rather than applicative: interpolation provides a controlled probe of what information the codec has organized into its latent space. If the representation is locally smooth and structured, simple geometric paths between two latent trajectories should decode into coherent audio, and the intermediate states can be analyzed to reveal regularities (e.g., rhythmic periodicity, channel coupling, and pitch dependence). This thesis therefore treats track-to-track interpolation as both (i) an audio-transition operator in a compressed domain and (ii) an experimental method for investigating the structure of codec latents.

1.1 Objective

The primary objective is to evaluate whether frame-wise interpolation between two bar-aligned tracks in a pretrained neural codec latent space can produce transitions that are perceptually and spectrally competitive with a conventional waveform-domain baseline (an equal-power crossfade under the same alignment and fade window).

The secondary objective is to characterize the latent representation used by the codec in the specific regime relevant to interpolation. The aim is to answer questions such as: How does an interpolated trajectory move between two latent timelines? Do diagnostics suggest tempo-linked periodicity? Are some latent channels systematically associated with particular frequency regions? How sensitive is the representation to pitch changes, and how do latent magnitudes relate to decoded loudness? These questions are addressed using latent-space trajectory analyses and targeted probing experiments alongside transition-quality evaluation.

1.2 Contribution

This thesis presents an EnCodec-based pipeline for bar-aligned latent interpolation between songs. It also implements baseline transition methods for comparison, including equal-power crossfades and pitch-preserved tempo-warping. It provides latent-space analyses of the transition behavior, such as correlation structure, principal component analysis (PCA) trajectories, and per-channel frequency contributions. This also p-rovides a clue to whether the latent space is organized in a way that can be exploited. Finally, it defines an evaluation setup that combines coherence analysis with subjective listening tests to assess perceptual quality.

1.3 Related Work

Interpolating between two points in a learned latent space is a common way to probe whether the representation is smooth and usable. This idea is well established in image models, where latent interpolation is used both as a sanity check and as a simple editing tool (Kingma and Welling, 2013; Karras et al., 2019). In music, latent interpolation has been explored in symbolic-sequence models such as MusicVAE (Roberts et al., 2018), and in VAE-based audio latent spaces in more exploratory/creative settings (Tatar et al., 2023). However, there appears to be limited published work that evaluates frame-wise interpolation directly in the latent *timeline* of neural audio codecs (e.g., EnCodec) as the main operator between two tracks. This thesis focuses on that codec-latent setting, and relates it to adjacent work below.

1.3.1 Music Models over Audio Tokens

Many neural music systems focus on continuation or prompt-based creation, rather than explicitly blending two fixed tracks. Jukebox is an early raw-audio approach that produces music via hierarchical tokenization and autoregressive modeling (Dhariwal et al., 2020). More recent work often models *discrete audio tokens*. AudioLM treats audio tokens as a sequence to be modeled, enabling continuation from short audio prompts (Borsos et al., 2022). MusicLM

extends this idea to text-conditioned music with longer-range structure (Agostinelli et al., 2023). MusicGen is particularly relevant here because it operates over EnCodec-style token streams, making the codec space a practical interface for modeling and control (Copet et al., 2023).

For transitions, the key point is that these models are not built around a "two-ended" constraint by default. This thesis therefore starts from a simpler operation: interpolate directly in a fixed codec latent space, and compare against waveform crossfades.

1.3.2 Autoencoders, Codecs, and Latent Manipulation

Vector-quantized autoencoders such as vector quantization - variational autoencoder (VQ-VAE) introduced discrete bottlenecks that support compression and token modeling (van den Oord et al., 2017a). Neural audio codecs build on this idea for high-fidelity reconstruction at low bitrate. SoundStream uses residual vector quantization (RVQ) to produce discrete codes (Zeghidour et al., 2021a), and EnCodec provides high-fidelity compression while exposing both continuous latents and quantized representations (Défossez et al., 2022).

A useful property of codec latents is that they form a *latent timeline*: a sequence of frames over time. This makes interpolation a natural, lightweight operator for transitions, since blending can be applied frame-wise over a fade window. The focus of this thesis is exactly this setting, using EnCodec as the codec backbone (Défossez et al., 2022).

Recent work also proposes music-focused codec tokenizers. HeartMuLa introduces *HeartCodec*, a low frame-rate music codec tokenizer aimed at preserving long-range musical structure while keeping token sequences short (Yang et al., 2026). This is closely related to the thesis viewpoint that the codec space itself is the main working representation, but as the work was published in the latter stages of this thesis, I did not have the time to investigate this properly.

1.3.3 Diffusion and Inpainting for Audio Editing

Diffusion models are another path for audio editing, as a transition region can be treated as "missing" audio that is filled in using context. DiffWave demonstrates diffusion for waveform synthesis (Kong et al., 2021), while AudioLDM and Moûsai apply latent diffusion to text-to-audio and text-to-music settings (Liu et al., 2023; Schneider et al., 2023). These approaches are promising for future refinement, but are more complex than the fixed-codec interpolation studied here.

1.3.4 Applications in Music Transitions and Blending

In DJ automation, transition quality depends strongly on choosing good regions and aligning beats/bars before any blending (Bittner et al., 2017). Other work learns transition parameter trajectories with differentiable effects (Chen et al., 2022), and cue-point detection methods aim to find good switch locations (Zehren et al., 2020; Argüello et al., 2024). But rather than learning effect controls in waveform space, this thesis asks whether a codec latent space (EnCodec) can serve as a clean domain for transition blending under fixed bar alignment.

Chapter 2
Theoretical Background

2.1 Digital Audio Representation

Audio in the real world is a continuous pressure waveform. A microphone converts this pressure variation into an analog electrical signal that changes smoothly in time and amplitude. A digital system cannot store a continuous signal directly, so it must measure the signal at discrete times and store each measurement as a finite number. If we measure too slowly, high-frequency content folds back into lower frequencies, and if we store each sample too coarsely, we introduce rounding errors. These two constraints, sampling and quantization, are the fundamental limits of what the digital signal can represent (Oppenheim and Schafer, 2010; Pohlmann, 2010).

Most signal processing in modern music and audio is performed on the digital representation, since it is easier to store and manipulate than its analog counterpart. Because the focus is music, the discussion assumes standard formats such as stereo pitch-code modulation (PCM) at 44.1 kHz/16-bit for distribution and 48 kHz/24-bit for studio work. These formats cover the audible range and provide headroom for processing (Pohlmann, 2010). The following subsections describe these building blocks of digitized audio and their practical consequences.

2.1.1 Sampling and Quantization

Sampling converts a continuous-time signal $x(t)$ into a discrete-time sequence $x[n] = x(nT)$ with sampling period $T = 1/f_s$. The sampling theorem states that if the signal is band-limited, it can be perfectly reconstructed from samples taken above twice its highest frequency (Oppenheim and Schafer, 2010). Real audio is not perfectly band-limited, so an analog anti-aliasing filter is used before conversion to suppress content above the Nyquist frequency $f_s/2$ (Oppenheim and Schafer, 2010).

Quantization then maps each sample to one of 2^N discrete levels for an N-bit system. This rounding introduces an error that is often modeled as additive noise whose power decreases as bit depth increases (Pohlmann, 2010). In practical terms, bit depth sets a limit on low-level detail and the cleanliness of quiet tails or fades. While quantization noise is often masked by musical content, it can become perceptible after multiple processing stages or when low-energy material is later amplified.

2.1.2 Frame-Based Processing and Windowing

Framing means splitting a discrete signal $x[n]$ into short segments of length L, typically with overlap determined by a hop size H. Windowing means multiplying each frame by a tapering function $w[n]$ (e.g., Hann or Hamming) so the segment fades toward zero at its boundaries. This reduces discontinuities and controls spectral leakage (Oppenheim and Schafer, 2010). Many audio operations are performed on these short, overlapping frames rather than on the full waveform. Window choice and frame length influence transient preservation and leakage, while hop size affects temporal smoothness and computational load (Oppenheim and Schafer, 2010).

Windowing also interacts with boundary selection. If a segment is cut without regard to phase or energy continuity, the splice can introduce a click that is unrelated to the processing method itself. This makes framing choices part of the baseline signal-processing assumptions used throughout the thesis.

2.1.3 Time-Frequency Analysis

Musical signals are highly non-stationary, so their spectral content changes over time. This property makes it less meaningful to compute the frequency content of a whole song. The short-time Fourier transform (STFT) addresses this problem by computing spectra on overlapping windows instead, trading time resolution against frequency resolution via window length and hop size (Allen and Rabiner, 1977; Oppenheim and Schafer, 2010). A common discrete-time definition is

$$X(m, \omega) = \sum_{n=-\infty}^{\infty} x[n]\, w[n - mH]\, e^{-j\omega n}, \tag{2.1}$$

where $x[n]$ is the discrete-time signal, $w[\cdot]$ is the analysis window, m is the frame index, H is the hop size in samples, n is the sample index, and ω is the digital angular frequency (typically in $[-\pi, \pi]$).

This representation underlies spectrograms and many objective measures of spectral change. For music analysis, the STFT is useful because it makes it possible to track how energy moves across frequency bands during different segments.

In many applications, the linear-frequency spectrogram is mapped to a mel-spectrogram, where the frequency bins are spaced on a perceptual scale rather than uniformly in Hertz. The main motivation is that music (and hearing) does not treat all frequency regions equally.

Low frequencies benefit from finer resolution to capture fundamental and closely spaced harmonics, while higher frequencies can be represented more coarsely without losing the overall spectral balance (Pohlmann, 2010; Müller, 2015; Stevens et al., 1937). A mel-spectrogram is obtained by projecting STFT power onto a mel filterbank. Let $|X(m,k)|^2$ denote the STFT power at frame m and FFT bin k, and let $M_{b,k}$ be the weight of the b-th mel filter at bin k. The mel-band energy is then

$$S_{\text{mel}}(m,b) = \sum_k M_{b,k} \, |X(m,k)|^2,$$

often followed by log compression to reduce dynamic range,

$$\tilde{S}_{\text{mel}}(m,b) = \log(\epsilon + S_{\text{mel}}(m,b)).$$

Compared to evenly spaced FFT bins, this representation reduces dimensionality and smooths fine bin-to-bin variation while preserving the broader spectral shape that is typically most relevant for perceived timbre and harmonic content (Slaney, 1998).

2.1.4 Reconstruction Artifacts

Even when the sampling theorem is respected, practical systems introduce artifacts. Resampling or time-scale modification can create aliasing if high-frequency content is not filtered properly, and windowed processing can cause ringing or temporal smearing around sharp events (Oppenheim and Schafer, 2010; Pohlmann, 2010). These effects are especially noticeable near window boundaries and in short segments.

Understanding the limitations of digital representation helps separate artifacts introduced by preprocessing from changes caused by later processing steps. This distinction is useful for interpreting results in the later analysis.

2.2 Music Theory Background

Music is more mathematical and predictable than one might expect. In everyday listening, patterns in time and pitch are often easy to hear. The main rhythm, regular groupings of beats, and the tonal center that makes certain notes feel "stable" all have some mathematical connection (Temperley, 2001; Lerdahl and Jackendoff, 1983). Music theory names these patterns tempo, meter, bars, phrases, key, and harmony, which together provide a compact way to describe structure in musical signals (Müller, 2015; Temperley, 2001; Lerdahl and Jackendoff, 1983). At a basic level, rhythm is built on periodic timing and harmony is based on consistent frequency relationships between notes, which makes musical structure measurable (Müller, 2015).

2.2.1 Musical Structure

Music is often organized in layers of time. Beats form bars, bars form phrases, and phrases form larger sections such as intro, verse, chorus, bridge, and outro (Lerdahl and Jackendoff,

1983). Repetition creates a sense of expectation, so listeners can anticipate when changes are likely to happen (Huron, 2006). Musicians often deviate or create variations of the main repetition during a song, as it adds further excitement to the track. Lots of pop-music and dance tracks follow regular phrase lengths, commonly in multiples of four bars of four beats, which makes the musical structure predictable in both listening and performance contexts (Butler, 2006). These boundaries are typically accompanied by changes in instrumentation, texture, dynamics, or harmony. Such changes are audible for listeners but also visually in time-frequency representations, and they provide cues for segmenting music into meaningful parts (Müller, 2015). Later sections make use of these cues for segment selection and alignment.

2.2.2 Tempo, Beats and Bars

Tempo describes the rate of the beat, typically measured in beats per minute (BPM). The beat itself is a perceptual pulse, such as a drum kick, which listeners and dancers can synchronize to. The synchronization of the beat is why song tempo is central in danceable music and DJ practice (Butler, 2006). Metering groups beats into repeating patterns of strong and weak positions within a bar. The downbeat marks the beginning of the bar and might be called strong, as it often carries an extra perceptual accent that sets the rhythmic structure (Müller, 2015; Temperley, 2001; Lerdahl and Jackendoff, 1983). Tempo can be steady throughout the track, vary between sections or vary locally through microtiming deviations (Müller, 2015). However, top-charting music mostly follow a constant tempo throughout the track.

2.2.3 Key, Tonality and Harmonic Compatibility

A musical key can be described in frequency terms. The key consist of a reference frequency (the tonic) plus a set of other frequencies (notes) that relate to it by fixed intervals, which defines the scale in which it is in. In Western music, the most common tuning system is 12-tone equal temperament (12-TET), where the pitches are spaced by a ratio of $2^{1/12}$. Most Western tonal music then uses diatonic major or minor scales, which are 7-note subsets of the 12-TET pitch set (Temperley, 2001; Krumhansl, 1990). The intervals we call "consonant" are close to simple ratios such as octave 2:1, fifth 3:2, fourth 4:3, and major third 5:4. Because musical notes contain harmonic overtones, these simple ratios line up many of those overtones and reduce beating (the slow amplitude wobble caused by interference between nearby frequencies), making the intervals sound more pleasant (Sethares, 2005; Plomp and Levelt, 1965). Tonality describes why some notes and chords feel stable while others create tension that resolves back to the tonic (Krumhansl, 1990; Temperley, 2001).

Harmonic compatibility between two segments depends on how similar their tonal centers and pitch collections are. When keys share many scale tones, the blend tends to feel natural, but when they share fewer tones, it can feel more abrupt and unpleasant (Krumhansl, 1990; Temperley, 2001). This gives a simple theory-based way of reasoning about the harmonic fit between two tracks.

2.3 Conventional Transition Techniques

Conventional transitions in music production and DJ practice rely on signal-processing methods that are well understood and easy to control. These techniques do not generate new material. Instead, they use timing and mixing so that two tracks overlap in a seamless way. The most common tools are beatmatching, time-stretching, and crossfades. Experienced producers or DJs sprinkle extra flair by using more advanced tools such as frequency filtering, along with synthetic effects like reverb and echo.

2.3.1 Beatmatching and Tempo Alignment

Beatmatching is all about aligning the periodic structure of two tracks so their beats and bars line up. In practice, this involves estimating tempo and beat positions, adjusting playback speed to match BPM, and aligning downbeats to minimize rhythmic conflict (Müller, 2015). This is often performed first when preparing a transition between two tracks. When executed well, the listener should not recognize a strict change of tempo or rhythmic structure during the transition. In DJ contexts beatmatching is done by ear, but often with the aid of algorithmic beat recognizers providing visual clues.

In modern DJ equipment it is also possible to synchronize the tracks BPM so they match, and at the same time be connected to one same tempo switch. This opens up the possibility of switching cleanly from track A's original tempo to track B's tempo by BPM ramping, while keeping the rhythm synchronized. This is more technical, but a common trick used by genre-fluid DJs.

2.3.2 Time-Stretching Methods

Time-stretching changes the duration of a signal without shifting its perceived pitch. Approaches such as overlap-add (OLA) and waveform-similarity OLA operate directly on short waveform frames and aim to preserve local waveform structure. Another method called the phase vocoder operates on the frequency-domain instead, modifying STFT frames and aims to maintain phase continuity across time (Zölzer, 2011; Laroche and Dolson, 1999). Each approach trades off transient sharpness, phase, and computational cost, and these artifacts can become audible when stretching factors are large (Zölzer, 2011). Time-stretching is a very important tool for BPM-modification, which is crucial for beatmatching. The specific backends used in this thesis are described later in the Method chapter.

2.3.3 Crossfades and Equal-Power Blending

A crossfade blends two signals by decreasing the gain of the outgoing track and an increasing gain to the incoming track over an overlap region. A standard linear crossfade uses linear gain curves, whereas an equal-power crossfade uses nonlinear curves that keep the sum of squared gains closer to constant. This tends to avoid a dip in energy for correlated signals (Zölzer, 2011). Crossfades are a simple method of switching from track A to track B, as they basically only act on the volume control of the tracks.

2.3.4 Adhering to Musical Structure

There are also less technical aspects of performing a transition as a DJ. The first and most important step is to pick tracks that fit in energy and section type. Harmonic fit helps but does not tell the whole story, since perceived energy is a crucial part of keeping a dancefloor alive. Finding a fitting transition is therefore as much about musical context as it is about technical alignment. Because dance music often uses regular phrase lengths, entry and exit points are commonly planned at bar or phrase boundaries, which makes the change feel expected rather than sudden (Butler, 2006). The technical steps are first to match tempo so the beats can line up, then align downbeats and bar boundaries to keep the rhythm coherent and minimize structural disruption. With this main structural concern resolved, the next step is the actual mix or track switch. This job is often done with a crossfade, or some variation of it using EQ or effects to manage energy while the underlying structure remains aligned.

2.4 Neural Networks and Backpropagation

Machine learning is based on the idea that a model can learn patterns from data instead of being programmed with fixed rules. In practice, we define a parameterized function $f(\mathbf{x};\theta)$ and fit its parameters θ by minimizing a loss over a dataset. In supervised learning the targets are provided, while in unsupervised learning the model searches for structure in the data itself. The central goal in both cases is generalization, meaning good performance on unseen data, not only on the training set (Russell and Norvig, 2020; Goodfellow et al., 2016).

A typical supervised objective is

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell\Big(f(\mathbf{x}^{(i)};\theta), \mathbf{y}^{(i)}\Big), \tag{2.2}$$

where each $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ is an input–target pair and N is the number of training examples. The loss $\ell(\cdot,\cdot)$ measures how far the prediction $f(\mathbf{x}^{(i)};\theta)$ is from the target $\mathbf{y}^{(i)}$. Minimizing the average loss encourages the model to fit the data while avoiding solutions that only work for a few samples. The exact form of ℓ depends on the task. For regression it is often a squared error, while for classification it is commonly cross-entropy (Goodfellow et al., 2016).

2.4.1 Neural Network Building Blocks

A neural network applies repeated affine transforms followed by nonlinearities. For an input vector $\mathbf{x}$, a layer maps

$$\mathbf{h} = \sigma(W\mathbf{x} + \mathbf{b}), \tag{2.3}$$

where $\sigma(\cdot)$ is an activation function such as ReLU or tanh. Stacking such layers yields a deep network that can learn progressively richer representations (Goodfellow et al., 2016). A short derivation of feed-forward layers and backpropagation is given in Appendix B.

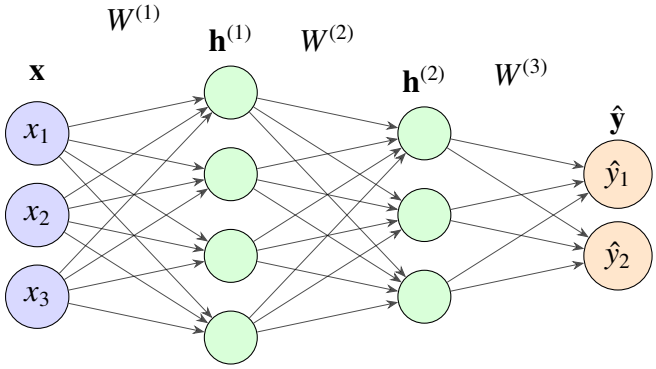

Figure 2.1: A feed-forward network with two hidden layers. Colors indicate layer types and $W^{(l)}$ denotes the weight matrix between layers.

2.4.2 Convolutional Layers

Convolutional layers are well suited for audio because they capture local structure and reuse the same filters across time. A 1D convolution for a single channel can be written as

$$y[n] = \sum_{k=0}^{K-1} h[k]\, x[n-k], \tag{2.4}$$

where $h[k]$ is a kernel of length K. Each kernel acts as a learned pattern detector, and stacking layers lets the model represent longer-range structure. Using a stride $s > 1$ downsamples the signal and reduces temporal resolution, trading detail for a broader context window (LeCun et al., 1998; Goodfellow et al., 2016).

After several convolutional layers, the feature maps are often pooled or flattened and passed through a fully connected layer, or a small multi-layer perceptron (MLP) to produce the final output representation or prediction (LeCun et al., 1998; Goodfellow et al., 2016).

2.4.3 Recurrent Layers (LSTM)

Recurrent layers model temporal context by carrying a hidden state forward in time. Vanilla recurrent neural networks (RNN) struggle with long sequences because gradients can vanish or explode during backpropagation through time, so earlier information is often forgotten. Long-short term memory (LSTM) mitigate this by introducing a gated cell state that preserves information over longer spans and stabilizes training (Hochreiter and Schmidhuber, 1997). In practice, this makes them useful when sequence context matters over many time steps. A more detailed LSTM update is included in Appendix B.

2.5 Neural Audio Codecs and Latent Spaces

Traditional audio codecs such as MP3 and AAC follow an algorithmic pipeline. First, the input (audio waveform) is transformed to remove redundancy, a psychoacoustic model estimates what can be discarded, and quantization plus entropy coding (huffmann/arithmetic)

yields a compact bitstream. Neural audio codecs however learn the analysis–synthesis transform directly from data using backpropagation through the neural net.

Neural codecs are often framed as autoencoders because the encoder–decoder structure mirrors compression. They map a waveform $\mathbf{x}$ to a lower-rate bottleneck and reconstruct it. The autoencoder becomes a *codec* only when that bottleneck carries limited information and can be expressed as bits, meaning it requires a discrete space.

For the remainder of this thesis, *latent* refers to the compact bottleneck representation produced by the encoder. In learned codecs it appears as the continuous latent $\mathbf{z}$ (raw encoder output), the quantized latent $\mathbf{z}_q$ (a real-valued reconstruction from codebook vectors), and the discrete code indices used for storage. These distinctions matter for interpolation as continuous or quantized latents can be blended smoothly, while discrete codes cannot.

2.5.1 Autoencoders

An autoencoder consists of an encoder E and a decoder G (Hinton and Salakhutdinov, 2006). Given a waveform $\mathbf{x} \in \mathbb{R}^{C\times T}$ with C channels and T samples, the encoder maps $\mathbf{x}$ to a lower-rate latent sequence,

$$\mathbf{z} = E(\mathbf{x}), \qquad \mathbf{z} \in \mathbb{R}^{D\times T'}, \tag{2.5}$$

where D is the number of latent channels and T' is the number of latent time steps after downsampling. The decoder then reconstructs

$$\hat{\mathbf{x}} = G(\mathbf{z}). \tag{2.6}$$

If the bottleneck is in a continuous space, the model is a learned representation but not yet a practical codec, because storing floating-point latents is expensive and does not provide a clear, discrete bitrate control.

2.5.2 Vector Quantization

Compression thus requires a discrete form of storing data. A common approach is vector quantization (VQ), popularized in VQ-VAE models (van den Oord et al., 2017b). Let $\mathcal{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_K\}$ be a learned codebook with $\mathbf{e}_k \in \mathbb{R}^D$. For each time step t, the encoder output $\mathbf{z}_t \in \mathbb{R}^D$ is mapped to the nearest codebook entry:

$$k_t = \arg\min_{k\in\{1,\ldots,K\}} \|\mathbf{z}_t - \mathbf{e}_k\|_2, \qquad q(\mathbf{z}_t) = \mathbf{e}_{k_t}. \tag{2.7}$$

The discrete index k_t is what can be stored, which is roughly $\log_2 K$ bits per step before entropy coding, while the decoder receives the embedded vector $q(\mathbf{z}_t)$. With a latent frame rate f_{latent}, the raw rate is approximately $R \approx f_{\text{latent}} \log_2 K$ bits/s for a single codebook, scaling linearly if multiple codebooks are used.

Because the nearest-neighbor assignment is non-differentiable, VQ models typically use a straight-through estimator during training and a commitment loss that keeps encoder outputs close to selected codebook vectors (van den Oord et al., 2017b). EnCodec uses residual vector quantization (RVQ), which stacks multiple codebooks; this is described in the next subsection (Défossez et al., 2022).

2.5.3 EnCodec

EnCodec is a learned neural audio codec with an encoder–quantizer–decoder structure, trained end-to-end with a combination of reconstruction losses and adversarial/perceptual losses (discriminators). EnCodec maps waveform audio $\mathbf{x}$ to a downsampled latent sequence $\mathbf{z}$, quantizes it with RVQ into discrete codes $\mathbf{k}$, and reconstructs $\hat{\mathbf{x}}$ from the summed codebook embeddings $\mathbf{z}_q$. Bitrate is controlled by the number of RVQ codebooks N_q. This thesis explores the latent space of the official pre-trained EnCodec described below to achieve maximum fidelity.

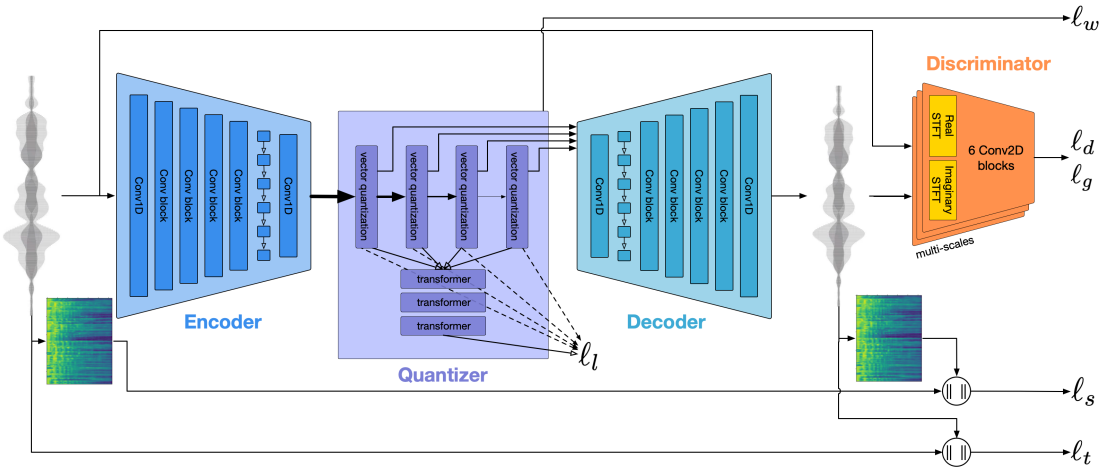

Figure 2.2: Overview of the EnCodec neural audio codec. An encoder maps waveform segments to a latent representation, which is quantized by a residual vector quantizer (RVQ) to produce a sequence of codebook indices. A decoder reconstructs the waveform from these discrete tokens. Adapted from Défossez et al. (2022).

Figure 2.2 illustrates the overall structure of EnCodec. The key idea is to learn an analysis–synthesis pipeline where the continuous encoder output is converted into a compact discrete representation via residual vector quantization. Using multiple quantizer stages allows the model to trade off bitrate and reconstruction quality by selecting how many codebooks to transmit, while the decoder remains the same. This makes EnCodec a practical choice for neural audio compression and a convenient source of discrete latent tokens for downstream generative modeling (Défossez et al., 2022).

Encoder/decoder and temporal downsampling

The encoder is a 1D convolutional stack with four downsampling stages with strides $(2, 4, 5, 8)$, followed by a two-layer LSTM for sequence modeling in the latent domain, and a final projection to a D-dimensional latent. The decoder mirrors the encoder using transposed convolutions. The total downsampling factor is the product of strides,

$$H = 2 \cdot 4 \cdot 5 \cdot 8 = 320 \text{ samples per latent step.}$$

This implies a latent frame rate

$$f_{\text{latent}} = \frac{f_s}{H},$$

so the paper's reported rates are 75 latent steps/s at $f_s = 24\,\text{kHz}$ and 150 at $f_s = 48\,\text{kHz}$, meaning one latent step corresponds to about 13.3 ms at 24 kHz and 6.7 ms at 48 kHz.

This motivates a *latent timeline*, where if n is a waveform sample index, the corresponding latent index is approximately $t \approx \lfloor n/H \rfloor$, i.e.,

$$\text{time}(t) \approx \frac{t}{f_{\text{latent}}}.$$

Discrete code indices (RVQ)

In vector quantization, each continuous latent vector is replaced by (i.e., projected onto) the nearest entry in a finite codebook (2.7). Residual vector quantization (RVQ) refines this idea by applying multiple codebooks sequentially. This means that after quantizing with the first codebook, it computes the residual error and quantizes that residual with a second codebook, and so on (Zeghidour et al., 2021b).

RVQ converts $\mathbf{z}$ into discrete code indices

$$\mathbf{k} \in \{1, \ldots, K\}^{N_q \times T'},$$

where K is the codebook size and N_q is the number of codebooks used for the target bandwidth. For each time step t, this can be written as

$$\mathbf{r}_0(t) = \mathbf{z}(t), \qquad \mathbf{q}_i(t) = Q_i(\mathbf{r}_{i-1}(t)), \qquad \mathbf{r}_i(t) = \mathbf{r}_{i-1}(t) - \mathbf{q}_i(t), \qquad \mathbf{z}_q(t) = \sum_{i=1}^{N_q} \mathbf{q}_i(t).$$

At decode time, $\mathbf{k}$ is mapped to codebook embeddings and summed across codebooks to form $\mathbf{z}_q$, which is then passed to the decoder to reconstruct $\hat{\mathbf{x}}$. in

Operational modes: non-streamable chunking vs. streamable inference

EnCodec is configured with two variants; a high-fidelity non-streamable setup operating at 48 kHz and a low-latency streamable setup operating at 24 kHz (Défossez et al., 2022). The architectural backbone (strided convolutions, LSTM over latents, and transposed-convolution decoder) is shared, but padding, normalization, and state handling differ to meet the latency constraint. As this thesis main focus is on the perceptual quality achievable from latent interpolation, the non-streamable setup at 48 kHz is investigated more thoroughly due to higher fidelity.

In the non-streamable configuration, convolutions use symmetric padding, and the input waveform is split into 1 s chunks with a 10 ms overlap to avoid boundary clicks. Each chunk is scaled (amplitude normalized) before encoding, and an inverse scaling is applied after decoding. The corresponding scale must therefore be transmitted as side information (Défossez et al., 2022). Because chunking introduces an explicit overlap region, implementation must also choose a stitching rule, either overlap-add or crossfading, when recombining decoded chunks.

In the streamable configuration, the model is modified for causal, low-latency operation by shifting padding and handling decoder outputs incrementally so audio can be decoded as soon as latent frames become available (Défossez et al., 2022).

Training objective

EnCodec is trained end-to-end by optimizing reconstruction losses both in time and spectrogram domains, together with an adversarial perceptual term based on a multi-scale STFT discriminator, and an RVQ commitment term (Défossez et al., 2022). Using the loss notation of the paper 2.2, the full codec (encoder + quantizer + decoder) plays the role of the generator and is trained with

$$L_G = \lambda_t \, \ell_t(\mathbf{x}, \hat{\mathbf{x}}) + \lambda_s \, \ell_s(\mathbf{x}, \hat{\mathbf{x}}) + \lambda_g \, \ell_g(\hat{\mathbf{x}}) + \lambda_{\text{feat}} \, \ell_{\text{feat}}(\mathbf{x}, \hat{\mathbf{x}}) + \lambda_w \, \ell_w, \tag{2.8}$$

where ℓ_t is the time-domain (waveform) reconstruction loss, ℓ_s is the multi-scale mel-spectrogram loss, ℓ_g is the generator adversarial loss, ℓ_{feat} is the (relative) feature matching loss, and ℓ_w is the RVQ commitment loss (Défossez et al., 2022). The optional entropy model is trained with a language-modeling loss ℓ_l over the discrete codes which is not covered in this thesis, however the losses included in 2.8 are further explained below.

Reconstruction losses

The time-domain reconstruction term is an L1 loss on the waveform,

$$\ell_t(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_1. \tag{2.9}$$

In the frequency domain, EnCodec uses a *multi-scale* mel-spectrogram loss with 64 mel bins, computed from a normalized STFT at several resolutions (Yamamoto et al., 2020; Gritsenko et al., 2020). Let $S_i(\cdot)$ denote a mel-spectrogram using window length $L_i = 2^i$ and hop size $H_i = L_i/4$. With the scale set $\mathcal{S} = \{5, \ldots, 11\}$, the windows range from $2^5 = 32$ to $2^{11} = 2048$ samples (Défossez et al., 2022). The paper combines L1 and L2 distances (and sets $\alpha_i = 1$), giving

$$\ell_s(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left(\|S_i(\mathbf{x}) - S_i(\hat{\mathbf{x}})\|_1 + \alpha_i \, \|S_i(\mathbf{x}) - S_i(\hat{\mathbf{x}})\|_2 \right), \qquad \alpha_i = 1. \tag{2.10}$$

Small windows emphasize transient detail, while large windows provide finer frequency resolution and better capture harmonic structure. This term complements ℓ_t which in itself is lacking precision due to timing inconsistencies between the original and reconstructed waveforms.

Adversarial perceptual loss and feature matching

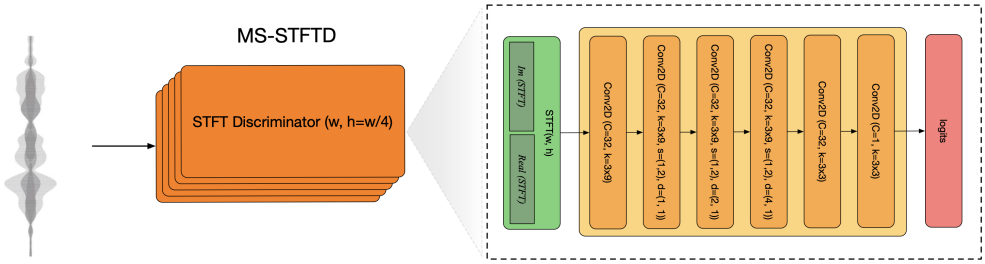

Figure 2.3: The discriminator architecture. Figure from Défossez et al. (2022).

To reduce artifacts, EnCodec uses a multi-scale STFT discriminator $\boldsymbol{D}$ (shown in Figure 2.3 that operates on complex STFTs (Défossez et al., 2022). Reasoning of using multiple STFT resolutions is to capture different dependencies in the structure or detail, similarly as for the reconstruction loss. Note that these discriminator STFT parameters are not the same as the reconstruction; at 48 kHz the multi scale - STFT (MS-STFT) discriminator uses five complex-STFT windows of sizes [4096, 2048, 1024, 512, 256] (Défossez et al., 2022). As the discriminator faces a classification task, it is trained with a hinge loss,

$$L_d(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{K} \sum_{k=1}^{K} \Big(\max(0, 1 - D_k(\mathbf{x})) + \max(0, 1 + D_k(\hat{\mathbf{x}})) \Big), \tag{2.11}$$

where k indexes the K STFT scales. In figure 2.2, the symbol ℓ_d refers to this discriminator hinge objective.

The generator receives the corresponding adversarial term,

$$\ell_g(\hat{\mathbf{x}}) = \frac{1}{K} \sum_{k=1}^{K} \max(0, 1 - D_k(\hat{\mathbf{x}})). \tag{2.12}$$

In addition, a relative feature-matching loss is used as a perceptual term,

$$\ell_{\text{feat}}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{KL} \sum_{k=1}^{K} \sum_{l=1}^{L} \frac{\|D_k^l(\mathbf{x}) - D_k^l(\hat{\mathbf{x}})\|_1}{\text{mean}\Big(\|D_k^l(\mathbf{x})\|_1\Big)}, \tag{2.13}$$

where D_k^l denotes the activation at layer l of discriminator k (Défossez et al., 2022). Because the discriminator can overpower the generator, its updates are probabilistically throttled (reported as $2/3$ at 24 kHz and 0.5 at 48 kHz) (Défossez et al., 2022).

RVQ commitment loss and multi-bitrate training

EnCodec adds a RVQ commitment loss ℓ_w that keeps the encoder outputs close to the selected codebook entries. The key detail is that the quantized value is treated as a constant so that no gradient is computed through the codebook lookup (Défossez et al., 2022). In the paper's

notation, for each residual quantization step $c \in \{1, \dots, C\}$ (where C depends on the chosen bandwidth), let $\mathbf{z}_c$ be the current residual and $q_c(\mathbf{z}_c)$ the nearest codebook entry. Then

$$\ell_w = \sum_{c=1}^{C} \|\mathbf{z}_c - q_c(\mathbf{z}_c)\|_2^2 . \tag{2.14}$$

EnCodec is trained for multiple bitrates by varying how many RVQ codebooks are used by choosing $C = N_q$ for the whole batch. The paper also reports improved quality when using a dedicated discriminator per bitrate, where for each batch one bitrate is selected and only the corresponding discriminator is evaluated and updated (Défossez et al., 2022).

Loss balancing

To stabilize training, EnCodec uses a *loss balancer* that normalizes the gradient contribution from each generator loss term that depends on the reconstructed output $\hat{\mathbf{x}}$ (Défossez et al., 2022). For a set of losses $\{\ell_i\}$, define $\mathbf{g}_i = \partial \ell_i / \partial \hat{\mathbf{x}}$, and let $\langle \|\mathbf{g}_i\|_2 \rangle_\beta$ be an exponential moving average of $\|\mathbf{g}_i\|_2$. The paper rescales each gradient as

$$\tilde{\mathbf{g}}_i = R \, \frac{\lambda_i}{\sum_j \lambda_j} \, \frac{\mathbf{g}_i}{\langle \|\mathbf{g}_i\|_2 \rangle_\beta}, \tag{2.15}$$

and backpropagates $\sum_i \tilde{\mathbf{g}}_i$ instead of $\sum_i \lambda_i \mathbf{g}_i$. With $\sum_i \lambda_i = 1$, each λ_i can be interpreted as the fraction of the total generator gradient coming from loss ℓ_i. The paper uses $R = 1$ and $\beta = 0.999$ (Défossez et al., 2022). The commitment loss ℓ_w is not included as it is not defined with respect to $\hat{\mathbf{x}}$ (Défossez et al., 2022).

Training data

The paper reports that the 24 kHz model is trained on speech, noisy speech, music, and general audio, while the 48 kHz model is trained on music only. (Défossez et al., 2022)

2.6 Latent Interpolation Methods

Latent interpolation is widely used in image generation, where a single latent vector controls an entire image. Blending between two vectors often yields a smooth visual morph in VAEs and GANs, and is commonly used to probe what the latent space has learned (Goodfellow et al., 2014; Radford et al., 2016). In audio, the representation is not one vector but a time-series of latent frames, so interpolation must be applied along a latent timeline rather than at a single point.

In this thesis, the main object is EnCodec's latent timeline and interpolating in this space. An input waveform $\mathbf{x}$ is mapped to a latent sequence $\mathbf{z}$ (which becomes $\mathbf{z}_q$ after RVQ) at a fixed latent frame rate $f_{\text{latent}} = f_s / H$ with $H = 320$ samples per latent step (Défossez et al., 2022). A transition between two tracks can therefore be described as a blending problem over latent frames instead of over waveform samples.

A key practical choice is *which* representation to interpolate. Discrete code indices $\mathbf{k}$ cannot be smoothly blended, while $\mathbf{z}$ and $\mathbf{z}_q$ are continuous and can be mixed frame-wise. In this section, interpolation is described primarily in the continuous encoder latent space $\mathbf{z}$. If a discrete representation is required after interpolation (e.g., to store the result at a fixed bitrate), one can re-quantize the interpolated latents with the same RVQ codebooks to obtain discrete codes and $\tilde{\mathbf{z}}_q$.

2.6.1 Linear Interpolation

A linear interpolation is a common choice when interpolating in latent spaces assumed to be Euclidean. Given two latent sequences of equal length, $\mathbf{z}^{(A)} \in \mathbb{R}^{D\times T'}$ and $\mathbf{z}^{(B)} \in \mathbb{R}^{D\times T'}$, the most direct method is a combination at each latent time step,

$$\tilde{\mathbf{z}}(t) = (1-\alpha(t))\,\mathbf{z}^{(A)}(t) + \alpha(t)\,\mathbf{z}^{(B)}(t), \qquad \alpha(t) \in [0,1]. \tag{2.16}$$

The interpolation schedule $\alpha(t)$ determines how quickly the representation moves from A to B across the transition region (e.g., a linear ramp). For a region of length n, the ramp will then include n linearly spaced $\alpha(t)$ values.

2.6.2 Spherical Linear Interpolation (SLERP)

Linear interpolation in Eq. (2.16) assumes Euclidean geometry. If latent vectors lie approximately on a constant-norm manifold, spherical linear interpolation can reduce radial shrinkage and keep the interpolation on a hypersphere. For each time step, let $\mathbf{u} = \mathbf{z}^{(A)}(t)$ and $\mathbf{v} = \mathbf{z}^{(B)}(t)$, and define

$$\theta = \arccos\left(\frac{\langle \mathbf{u}, \mathbf{v}\rangle}{\|\mathbf{u}\|_2\,\|\mathbf{v}\|_2}\right),$$

with interpolation

$$\mathrm{slerp}(\mathbf{u}, \mathbf{v}; \alpha) = \frac{\sin((1-\alpha)\theta)}{\sin\theta}\mathbf{u} + \frac{\sin(\alpha\theta)}{\sin\theta}\mathbf{v}. \tag{2.17}$$

In a sequence setting, SLERP can be applied per time step as

$$\tilde{\mathbf{z}}(t) = \mathrm{slerp}(\mathbf{z}^{(A)}(t), \mathbf{z}^{(B)}(t); \alpha(t)).$$

In practice, the benefit depends on whether $\|\mathbf{z}(t)\|_2$ is approximately stable. Otherwise, SLERP may not better match the latent distribution than standard linear blending.

2.6.3 Sigmoid/Eased Interpolation Curve

Abrupt changes in $\alpha(t)$ might introduce noticeable discontinuities in the decoded audio, especially around transients. A common remedy is to replace a linear ramp with a smooth sigmoidal schedule. Let $u \in [0,1]$ denote normalized transition time; then

$$\alpha(u) = \frac{\sigma(\beta(u-1/2)) - \sigma(-\beta/2)}{\sigma(\beta/2) - \sigma(-\beta/2)}, \tag{2.18}$$

where $\sigma(\cdot)$ is the logistic function and β controls the steepness. This schedule starts and ends at 0 and 1, and can be used directly in Eq. (2.16).

2.6.4 Latent Alignment and Scale Handling

Latent interpolation assumes that the frames are musically synchronized, as described in Section 2.3.1 for the synchronization of the waveforms. If $\mathbf{z}^{(A)}(t)$ and $\mathbf{z}^{(B)}(t)$ represent different rhythmic phases or local tempos, frame-wise blending may not reflect the intended musical alignment. Alignment can be handled before encoding (e.g., beatmatching and time-stretching) or by time-warping one latent sequence onto the other's timeline (e.g., resampling or piecewise-linear warping of $\mathbf{z}$). Because one latent step corresponds to H waveform samples, any alignment target expressed in samples or seconds can be mapped to latent indices via $t \approx \lfloor n/H \rfloor$.

Amplitude scale differences between the two sequences can also bias interpolation by introducing unintended gain changes. This can be addressed by normalizing the inputs or by interpolating a scale trajectory alongside the latents.

Chapter 3
Method

This chapter is ordered chronologically so the reader can follow the pipeline from input to output. Data preparation and shared transition steps are described first. The pipeline then splits into a baseline path for the equal-power crossfade, or an EnCodec path for the actual latent interpolation before moving to evaluation.

3.1 Data and Shared Preparation

3.1.1 Tracks and selection criteria

The evaluation uses dance-music tracks exported as uncompressed waveform (WAV) files. All source material is prepared in Ableton Live. Tracks are aligned to the bar grid, trimmed to whole bars, and exported so each file starts on a bar downbeat and ends exactly on a bar boundary. This keeps clips bar-exact and makes the bar indices later in the pipeline clear and consistent (see Section 2.2). The resulting files have a bar length that is an integer multiple of eight, which matches common dance-music phrasing and makes it easier to choose transition points that feel expected.

Tracks are picked for stable tempo, consistent meter (mostly 4/4), and between pairs a reasonable match in energy and key so the transition is sensible. Stable tempo avoids drift during beat alignment, and a consistent meter keeps the bar mapping reliable across tracks. The tracks are mainly dance-focused music due to the nature of the application, with BPM ranging between 105 and 145.

3.1.2 Shared preprocessing and transition preparation

During inference, each track is loaded, resampled to the model sample rate (48 kHz for the main experiments), and converted to the required channel count. BPMs can be provided

manually if known, or estimated using a fixed backend. The default pipeline uses Essentia's RhythmExtractor2013, while an optional flag switches to librosa's `beat_track`. If the user supplies the BPMs, estimation is bypassed entirely. Bar indices are then mapped to sample ranges to keep alignment consistent with the bar-based structure discussed in Section 2.2.

A bar-based fade window is selected in both tracks by the user. The fade window is isolated so optional pitch-preserving time stretch or tempo warp can be applied, and the processed window is then stitched back into the full track. After this step, the fade regions are aligned and ready for the pipeline split.

Tempo-warping and pitch

When BPMs differ, alignment is applied to the fade region before the transition to avoid beat drift. Two options are used. The pitch-preserving option time-stretches each fade to a shared duration while keeping pitch unchanged. The tempo-warp option ramps playback speed across the fade, which changes pitch but keeps rhythmic alignment. Both options use fixed backends, more specifically the library rubberband.

Let $r = \mathrm{BPM}_B/\mathrm{BPM}_A$. For tempo-warp alignment, the playback speed in track A is ramped linearly from 1 to r across the fade, while track B is ramped from $1/r$ to 1. This creates a smooth tempo bridge so the number of beats in the overlap matches on both tracks, while leaving the rest of each track unchanged.

Combining pitch preserve and tempo warp is also an alternative and is the most common setting in this thesis. That results in a gradual tempo ramp but implements it with pitch-preserving time stretching, so the tempo changes smoothly without shifting key. After this step, the two tracks have aligned fade regions and are ready for the next stage.

3.2 Transition Pipelines

This section starts from the end of the shared preparation and is split into two pipelines. One stays in the waveform domain throughout and serves as the baseline. The other proceeds into the EnCodec latent space, which is the main exploration of this thesis.

3.2.1 Baseline transition

The baseline keeps the aligned fades in the waveform domain throughout the process. The selected bar window is initially split into a prefix, a fade region, and a suffix for each track. Only the fade regions are blended.

Let $x_A[n]$ and $x_B[n]$ be the two fade signals over N samples, and let $\alpha[n] = n/(N-1)$ for $n = 0, \ldots, N-1$. The blended fade is

$$y[n] = g_A[n]\, x_A[n] + g_B[n]\, x_B[n], \tag{3.1}$$

with equal-power gains

$$g_A[n] = \sqrt{1 - \alpha[n]}, \qquad g_B[n] = \sqrt{\alpha[n]}. \tag{3.2}$$

This choice keeps $g_A^2[n] + g_B^2[n] = 1$ and therefore holds the total energy approximately constant when the two fades are correlated. The final output is assembled from the prefix of track A, the blended fade, and the suffix of track B. This baseline is the reference for the objective metrics and the listening tests.

3.2.2 EnCodec Latent Interpolation Pipeline

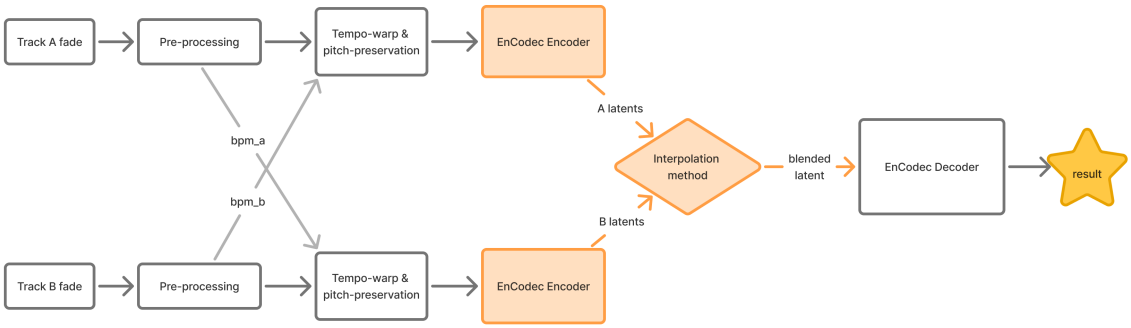

Figure 3.1: Simplified latent interpolation pipeline. This example excludes specific intermediate steps, such as slicing the waveforms into different parts, latent alignment etc.. The white boxes indicates it operates in waveform domain, while the orange indicate operations in or towards latent space.

Model configuration

The EnCodec variant and bandwidth settings define the latent frame rate, channel count, and bitrate. The main model is the 48 kHz non-streamable variant, while the 24 kHz streamable variant is also tested. The target bitrate is set per experiment and selects how many RVQ codebooks are active. If no bitrate is specified, the implementation returns to the highest available bandwidth of the model. All runs use the official pretrained checkpoints without finetuning, and audio is loaded with the model sample rate and channel count to avoid mismatches.

Latent extraction and normalization

Each track is encoded by the EnCodec encoder into a latent timeline. The 48 kHz non-streamable model uses chunked inference, so the waveform is split into overlapping segments and each segment is encoded separately. The segment length and stride come from the model configuration. In the official 48 kHz setup this means 1 s chunks with a 10 ms overlap. The overlap exists to avoid boundary clicks when chunked processing is used. In this implementation the latent segments and their scale envelopes are recombined with a linear overlap-add across that 10 ms region, matching the reference EnCodec utilities. The 24 kHz streamable model encodes the waveform in one pass and needs no recombination.

When normalization is enabled, each encoder segment is root mean square (RMS) - normalized to unit level before encoding. For the 48 kHz non-streamable model this is done per 1 s segment with 10 ms overlap, and for the 24 kHz streamable model it is a single pass over the whole track. The RMS is computed on a mono fold-down as $\sqrt{\text{mean}(x^2)}$, and a floor of 1×10^{-8} prevents division blow-ups on silence, meaning silent regions remain silent. The resulting scale factors are expanded across the latent frames (using the same overlap-add schedule) to form a scale envelope carried alongside the latents and re-applied after decoding.

Latent segmentation

After encoding, the bar-aligned sample window is mapped to latent frame indices using the fixed downsampling factor of the model. One latent frame corresponds to 320 samples, which is 150 latent frames per second at 48 kHz and 75 frames per second at 24 kHz. This mapping keeps the segmentation consistent with the bar-based structure discussed in Section 2.2.

The latent timeline is then split into prefix, fade, and suffix segments. Only the fade segment is used for interpolation, while the prefix from track A and the suffix from track B are preserved to maintain the original musical context around the transition.

Latent alignment

The fade segments from track A and track B must have matching lengths (frame count) before they can be blended. If the shared preparation already equalized the fade lengths in sample count, it means the latent frame counts should be equal as well. Latent alignment is still applied to guarantee identical frame counts after encoding, simply cropping to the shortest fade window of the two tracks.

When normalization is enabled, the scale envelopes are aligned alongside the latent fades so that loudness changes follow the same timing as the latent interpolation.

Interpolation and re-quantization

Interpolation is applied frame by frame to the aligned fade latents, using the methods described in the theoretical background. The blend is restricted to the fade window so the prefix and suffix remain unchanged. Each strategy defines how the interpolation schedule progresses across the fade, while keeping the endpoints fixed at the original latents.

Interpolation operates on the continuous encoder output z (pre-quantization). Discrete RVQ code indices are never averaged, but instead the blended z stays in a smooth space and can optionally be projected back to the codebook manifold after blending.

When normalization is active, the scale envelopes are blended linearly across the same fade window, independent of the latent interpolation method. This keeps loudness transitions smooth even when the latent path is nonlinear.

After interpolation, the blended latent can optionally be re-quantized with the same residual vector quantizer used by EnCodec, producing z_q before decoding. This enforces the target bitrate and keeps the decoded signal on the valid codebook manifold.

Latent assembly and decoding

After interpolation, the blended fade is stitched with the prefix from track A and the suffix from track B to form a complete latent timeline. This preserves the original context around the transition while replacing only the overlap region.

The assembled latent is decoded back to audio using the EnCodec decoder. For the 48 kHz non-streamable model, decoding follows the same 1 s / 10 ms overlap schedule as encoding and the latent (and scale envelope, when present) are recombined with a linear overlap-add across the overlap. The 24 kHz streamable variant decodes in one pass with no stitching. If normalization is enabled, the stored scale envelope is re-applied after decoding so the output retains the original loudness profile.

The pipeline can render either the full transition or only the fade region, depending on whether the prefix and suffix are included in the assembled latent. The resulting audio is written as a WAV file at the model sample rate (48 kHz for the main experiments, 24 kHz in the streamable ablations) with the model channel count (stereo in all runs).

3.3 Transition-quality evaluation

This section defines how the rendered transitions are assessed, with results presented in Chapter 4. Objective metrics are computed on the output audio using fixed analysis windows and hops, and AB listening tests are run on the same set with randomized order. Unless otherwise noted, each condition uses the same fade length (8 bars) and analysis sample rate (48 kHz). All of the summaries are reported in the results chapter, while this section specifies only the measurements.

3.3.1 Model selection for further analysis

Before running the full evaluation, a small pilot compared EnCodec variants and settings on a subset of transitions to choose a default configuration. The pilot tested ablations from both 24 kHz and 48 kHz models, using 6, 12, and 24 kbps bandwidth, and re-quantization on/off. Pitch-correction and tempo warping was also evaluated. For the interpolation itself, ablations of linear, SLERP and sigmoids were used. Quick AB or preference judgments focused on smoothness, artifacts, harmonic fit, and perceived energy. The configuration with the most consistent perceptual quality was selected for the main experiments (reported in Chapter 4), which was a setup using the 48 kHz model with 24 kbps bandwidth and re-quantization off. Interpolation method used was SLERP. The pre-processing used is pitch-correction and tempo-warp for both the latent interpolation and the baseline crossfade.

3.3.2 Listening tests

A custom web interface presents two audio samples (A and B) for each transition. One sample is produced by latent interpolation and the other by the baseline crossfade. The assignment of method to label (A/B) is randomized per trial to mitigate systematic presentation bias, such as a tendency to prefer the first-listed option or to associate a fixed label with a particular method. Participants listen to both samples and select the transition they prefer, with an additional *no preference* option to avoid forced choices when the two samples are perceived as equivalent.

Eight transition pairs are included in total, and each participant completes all 8 AB trials. The number of participants is $N = 14$. Responses marked as *no preference* are retained and reported as a separate outcome, meaning binary preference rates are computed from trials with a stated preference for A or B.

Statistical analysis. Responses marked as *no preference* are reported separately and excluded from the binary preference rate. Let n denote the number of decided judgments (excluding "no preference"), and let x denote the number of decided judgments in which SLERP is preferred. We report $\hat{p}_{\text{SLERP}} = x/n$.

To assess whether preferences differ from chance, we use an exact binomial test with $H_0 : p_{\text{SLERP}} = 0.5$ versus $H_1 : p_{\text{SLERP}} \neq 0.5$ (two-sided). For directional interpretation, we additionally report one-sided exact binomial p-values for the two possible alternatives: $H_1^+ : p_{\text{SLERP}} > 0.5$ (SLERP preferred) and $H_1^- : p_{\text{SLERP}} < 0.5$ (baseline preferred), using $\alpha = 0.05$.

3.3.3 Objective similarity metrics

Coherence

Objective evaluation uses time–frequency magnitude-squared coherence between the baseline crossfade and the latent transition. Let $x[n]$ be the equal-power crossfade over the fade region and $y[n]$ the rendered latent transition over the same region, both resampled to 48 kHz and folded to mono. The signals are optionally aligned by maximizing cross-correlation within ± 1 s before analysis. As the latent pipeline should not create large timing differences, we are mostly interested in frequency resolution. We compute STFTs with a Hann window of length $N_{\text{fft}} = 8192$ and hop $H = 2048$, then smooth the power spectra and cross-spectrum over a 1 s temporal window (Allen and Rabiner, 1977; Oppenheim and Schafer, 2010). The coherence is

$$C_{xy}(f,t) = \frac{|S_{xy}(f,t)|^2}{S_{xx}(f,t)\, S_{yy}(f,t)}, \tag{3.3}$$

where S_{xx} and S_{yy} are the smoothed auto-spectra and S_{xy} is the smoothed cross-spectrum. $C_{xy}(f,t) \in [0,1]$ measures how closely the transition follows the baseline in the time–frequency domain. We calculate mean coherence across all time–frequency bins and band-limited averages (default 0–200–2000–8000–20000). Frequencies between 20000 and Nyquist are excluded in the coherence analysis due to the EnCodec model's inability to reconstruct this region (see Section 3.3.4).

To characterize periodic structure in coherence as a function of frequency, we additionally compute the time-averaged coherence (mean coherence spectrum)

$$\bar{C}_{xy}(f) = \frac{1}{T} \sum_{t} C_{xy}(f, t). \tag{3.4}$$

After visual inspection of $\bar{C}_{xy}(f)$, there are indications of periodic structure between coherence peaks. We apply a simple peak sanity check where local maxima are detected, and peaks closer than 50 Hz are resolved by keeping the bin with higher coherence. From the remaining set, the top-N peaks (by coherence value) are reported, where N is chosen from the visual amount of peaks.

3.3.4 Qualitative audio diagnostics

Spectrogram comparisons

Per-bar spectrograms are generated for the baseline crossfade output and the EnCodec latent-interpolation output, alongside the corresponding fade regions from the original songs (track A and track B). All visualizations are restricted to the same bar-aligned fade window used in rendering and evaluation, so the four signals can be compared directly. To make musical phrasing explicit, the spectrograms are additionally segmented by bar and displayed as per-bar panels across the full fade length.

Unless otherwise stated, spectrograms use the same STFT configuration as the coherence analysis (Hann window, $N_{\text{fft}} = 8192$, hop $H = 2048$) to keep qualitative inspection consistent with Section 3.3.3. Magnitudes are displayed in dB with a fixed dynamic range across conditions for comparability.

3.4 Latent and diagnostic analyses

This section collects the analyses used to interpret both transition behavior and latent-space structure in EnCodec. The first blocks analyze rendered transition outputs in the latent domain. A dedicated pitch-coding block then uses controlled synthetic trials and single-track pitch shifts to isolate how fundamental frequency and overtone structure are reflected in latent representations. The final block links latent magnitude dynamics to decoded waveform power.

3.4.1 Latent trajectory diagnostics

Latent space

The latent space during the interpolation is first investigated through PCA. Here, PCA is applied to a data matrix whose rows are latent frames and whose columns are latent dimensions, formed by concatenating the frames from fade A, fade B, and the blended trajectory.

After mean-centering the columns, PCA is computed via the SVD of the centered matrix, $X = U\Sigma V^\top$. This is equivalent to eigendecomposition of the sample covariance matrix $\frac{1}{N-1}X^\top X$, but using the SVD directly is convenient and numerically stable. The principal directions are given by V, and the variance captured by component i is $\lambda_i = \sigma_i^2/(N-1)$, with explained-variance ratio $r_i = \lambda_i / \sum_j \lambda_j$ and cumulative ratio $R_k = \sum_{i=1}^{k} r_i$. In this thesis, trajectory plots use the first two PCs for visualization, while the full PCA spectrum is used when computing variance coverage, so we can quantify how much of the total latent variation is retained in the plotted subspace.

The 2D PCA plots are divided by bars, meaning 8 plots in total for a transition being 8 bars long. The interest is to investigate whether the tracks are moving around in different areas in the latent space. A better visualization of this is by animating the bar plots, so the individual points can be seen moving around in PCA space, which allows for a better view of how the latents behave in time.

To expose periodic structure in the latent motion, the time series of PCA scores along the trajectory is analyzed with a discrete time Fourier transform (DTFT). This yields dominant modulation rates for each component and is used to check whether tempo-related periodicity is visible in the latent dynamics, which can provide hints about the track's BPM or other temporal structures.

3.4.2 Latent structure diagnostics

Cosine similarity

Frame-wise cosine similarity is computed between the blended latent and each source latent. This yields two similarity curves over the fade that summarize how quickly the blend departs from track A and approaches track B, and can be averaged per bar using the same bar boundaries as the transition pipeline.

Cosine similarity is defined frame-wise between two latent vectors $z_A(t)$ and $z_B(t)$ as

$$c[t] = \frac{\langle z_A(t), z_B(t) \rangle}{\|z_A(t)\|_2 \, \|z_B(t)\|_2}, \tag{3.5}$$

and can be summarized using mean/min/max over time.

Correlation and Contribution

To show structure and redundancy in the latent channels, a channel-wise correlation matrix is computed across the transition pair tracks. This aggregates all latent frames and reports Pearson correlations between channels, giving a view of coupling or redundancy in the representation. Here, Pearson correlation measures linear association between two channels, taking values in $[-1, 1]$: positive values indicate that channels tend to vary together, negative values indicate opposite variation, and values near zero indicate weak linear dependence.

A frequency-contribution probe is used to see which channels carry which spectral content. Each latent channel is masked by zeroing the channel on both original tracks in latent space. The audio is then decoded, and the relative spectral loss is measured against the unmodified decode, computed on mel power spectra using fixed analysis settings. Optionally, a keep-only decode is used to visualize the standalone spectral footprint of each channel.

3.4.3 Pitch-coding diagnostics

This diagnostic block investigates whether pitch information (fundamental frequency and overtone structure) is represented systematically in EnCodec latent space. In contrast to the transition-pair diagnostics above, these trials use controlled synthetic inputs and single-track pitch shifts to isolate pitch-related effects. A single precomputed PCA basis from 64 dance tracks is reused across trials so trajectory plots stay directly comparable.

Controlled harmonic tone-pair trial

Two synthetic harmonic tones are generated with identical synthesis settings, differing only in fundamental frequency (e.g., A4 vs B4). This isolates pitch as the manipulated variable while keeping timbral synthesis parameters fixed (number of overtones, harmonic rolloff, envelope, duration, and peak level).

The two latent sequences are compared with frame-wise cosine similarity (Eq. 3.5), and summarized using mean/min/max over time. Both trajectories are also projected to the shared PCA basis for visual separation analysis.

Harmonic sweep trial

A harmonic sweep is synthesized from f_{start} to f_{end}, together with two constant references at the start and end frequencies. The sweep latent is compared to each reference over time, producing two cosine curves.

For interpretation, the cosine plot uses fundamental frequency on the x-axis (not latent-frame index), with explicit markers for start and stop frequency.

Track semitone-shift trial

A source song is pitch-shifted by +1 semitone (duration preserved), and original/shifted versions are encoded and compared. The trial reports global frame-wise cosine similarity and per-channel latent-change ranking.

For each channel i, the default ranking score is the normalized mean absolute change:

$$n_i = \frac{\frac{1}{T}\sum_t \left|z_i^{\text{shift}}(t) - z_i^{\text{orig}}(t)\right|}{\text{std}_t\left(z_i^{\text{orig}}(t)\right) + \epsilon}. \tag{3.6}$$

In addition, mean absolute difference and RMS difference are stored for the full ranking table.

PCA windowing for song-shift visualization

For readability, the song-shift PCA plot is restricted to a short musical window (default: one bar), while cosine and channel-ranking metrics are computed on the full latent sequence. The window length is controlled by bars, BPM, and beats per bar, and can be offset by a user-defined start time.

3.4.4 Latent magnitude vs decoded power

Latent space energy vs. decoded waveform power

To relate latent dynamics to output loudness, we compare a latent-energy signal to a decoded-power signal on the same timeline. Latent energy is represented by the frame-wise ℓ_2 norm:

$$\ell[t] = \|z(t)\|_2 = \sqrt{\sum_{i=1}^{D} z_i(t)^2}, \tag{3.7}$$

where $z(t) \in \mathbb{R}^D$ is the blended latent at frame t. This gives one scalar per frame that summarizes overall latent magnitude. Decoded waveform power is computed from the mono decoded waveform y_{mono}:

$$p[t] = \frac{1}{H}\sum_{n=0}^{H-1} y_{\text{mono}}^2(tH + n), \tag{3.8}$$

where H is the number of waveform samples per latent frame ($H \approx f_s/f_{\text{latent}}$). So $p[t]$ is mean-square power (RMS^2) per latent frame. If a small boundary mismatch occurs, both sequences are truncated to their common length before comparison.

Delay is then estimated with frame-lagged normalized cross-correlation between $\ell[t]$ and $p[t]$, after z-normalizing both sequences:

$$r_{\text{power,latent}}(k).$$

The peak is searched within approximately ± 1 second on the latent timeline:

$$k^\star = \arg\max_{|k| \le f_{\text{latent}}} r_{\text{power,latent}}(k),$$

and converted to milliseconds as

$$\tau = 1000\, k^\star / f_{\text{latent}}.$$

3.5 Implementation

3.5.1 Exact setups

All experiments are run via command line input (CLI) entry points, which wrap the preprocessing, interpolation, and analysis stages. The required inputs are the two audio file paths, bar indices for the fade region (1-based), the fade length in bars, and optional tempo information (either explicit BPMs or a fixed estimator backend). Model-specific inputs include the EnCodec variant (24 kHz or 48 kHz), optional bandwidth, interpolation mode, and whether to apply tempo warping, pitch-preserving stretch, or latent re-quantization. A minimal example of the latent interpolation call is shown below, and the full command listings (including baseline crossfade and analysis tools) are provided in the github repository which link is available in the Appendix D. The repository is public and a link to the codebase is also given there.

```
python scripts/encodec/interpolation/interpolate.py \
  --song-a data/raw/track_a.wav \
  --song-b data/raw/track_b.wav \
  --start-bar-a 9 \
  --start-bar-b 1 \
  --length-bars 8 \
  --mode linear \
  --tempo-warp \
  --output experiments/encodec/a_to_b.wav
```

3.5.2 Software Stack, Hardware

All experiments are run in Python with PyTorch, torchaudio, EnCodec, and librosa, with Essentia used optionally for BPM estimation. The exact package versions and the hardware configuration used for the reported results are listed here: Python `3.11.14`, PyTorch `2.9.1+cu130`, torchaudio `2.9.1+cu130`, EnCodec `0.1.1`, librosa `0.11.0`, rubberband `3.3.0`, Essentia `2.1-beta6-dev` (optional). Hardware used is AMD 9950X3D (CPU), RTX 5090 (GPU), and 96 GB of RAM.

3.5.3 Inference Settings and Runtime

The main experiments use the 48 kHz non-streamable EnCodec variant, with the 24 kHz streamable variant evaluated in ablations. Bandwidth is set per experiment to select the number of RVQ codebooks, and re-quantization is toggled explicitly when testing bitrate constraints. For each condition we report the interpolation mode, whether tempo warping and pitch-preserving stretch are enabled, and the average runtime per transition on the stated hardware.

Chapter 4
Results

This chapter reports results for the latent interpolation pipeline using the same track pairs and bar-aligned fade windows described in Method. Objective metrics, listening tests, and diagnostic plots are computed on the rendered outputs from the chosen EnCodec configuration at 48 kHz and 24 kbps bandwidth. The transition shown is between track A "You Are In My System (Kerri Chandler)" with a BPM of 128, and track B "What's a Girl To Do (Fatima Yamaha)" having a BPM of 120. This configuration enables the use of all possible pre-processing methods.

4.1 Transition-quality evaluation

4.1.1 Listening tests

Participants and Setup

A total of $N = 14$ participants completed the listening test, and all participants completed all eight AB trials ($14 \times 8 = 112$ judgments). Each trial compared a baseline equal-power crossfade to a latent interpolation transition (SLERP), with a *no preference* option available. The assignment of method to label (A/B) was randomized per trial.

Preference (AB) Results

Across all 112 judgments, the baseline was preferred 47 times (42.0%), SLERP was preferred 44 times (39.3%), and 21 trials (18.8%) were marked as *no preference*. Excluding *no preference* responses, SLERP received 44/91 decided preferences (48.4%) and the baseline received 47/91 (51.6%).

Table 4.1 breaks down preferences per transition pair. The strongest SLERP-leaning pair

in this dataset is `whitenoise_chances` (9 vs. 3 decided preferences), while the strongest baseline-leaning pair is `inmysystem_pursuit` (9 vs. 3 decided preferences).

Transition	Baseline	SLERP	No pref.	Decided	SLERP share (%)
`girl_whitenoise`	7	5	2	12	41.7
`hotel_cantgetyou`	4	7	3	11	63.6
`inmysystem_girl`	5	5	4	10	50.0
`inmysystem_lady`	8	5	1	13	38.5
`inmysystem_pursuit`	9	3	2	12	25.0
`pjanoo_pursuit`	5	7	2	12	58.3
`whitenoise_chances`	3	9	2	12	75.0
`whitenoise_hotel`	6	3	5	9	33.3
Overall	47	44	21	91	48.4

Table 4.1: AB preference counts per transition (N=14 participants, 8 trials each). "Baseline" is the equal-power crossfade and "SLERP" is the latent interpolation condition. "Decided" excludes "no preference".

An exact binomial test against $p_0 = 0.5$ found no evidence of a preference difference (two-sided $p = 0.834$). One-sided exact binomial tests likewise found no evidence that SLERP outperforms the baseline ($H_0 : p_{\text{SLERP}} \leq 0.5$, $p = 0.662$, $\alpha = 0.05$) or that SLERP is worse than the baseline ($H_0 : p_{\text{SLERP}} \geq 0.5$, $p = 0.417$, $\alpha = 0.05$).

Qualitative Comments

Five participants left at least one optional free-text comment, for a total of 14 non-trivial comments. The most common themes were timing/beat-alignment issues (e.g., "out of time" or "stuttering"), followed by mentions of artifacts such as distortion/clicks, and occasional reports of brief dropouts.

4.1.2 Objective similarity metrics

Coherence vs. Baseline

Figure 4.1 visualizes the time–frequency coherence between the baseline equal-power crossfade and the latent interpolation output for the representative pair.

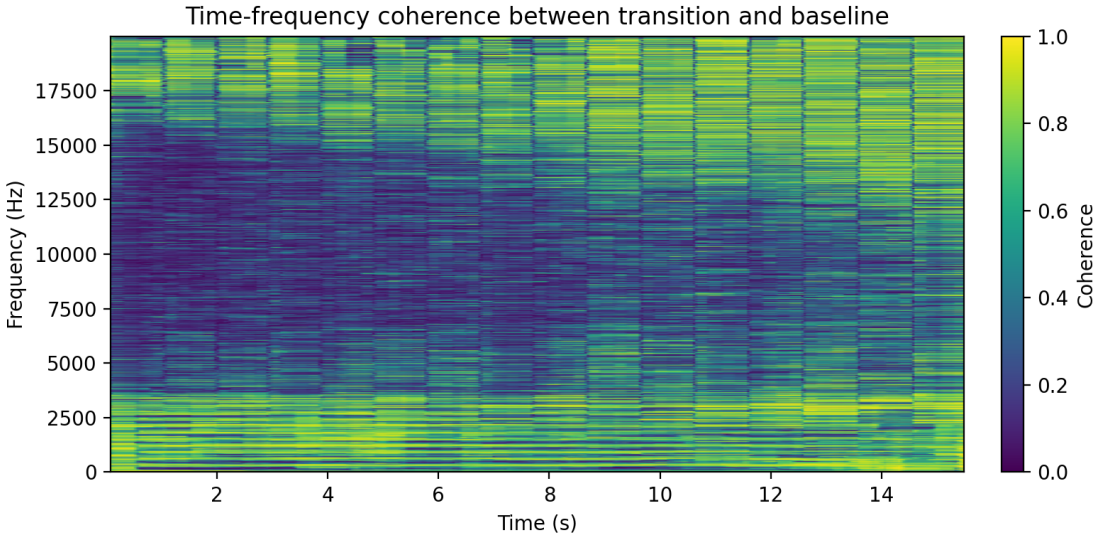

Figure 4.1: Time–frequency coherence between the baseline crossfade and the latent transition.

Band	Coherence
Overall	0.4340
0–200 Hz	0.4757
200–2000 Hz	0.6143
2000–8000 Hz	0.3941
8000–20000 Hz	0.4262

Table 4.2: Aggregate coherence values by band for the representative transition.

Because much musical content lies in the 0–4000 Hz region, and Figure 4.1 shows higher coherence there, this band is further evaluated on the next page.

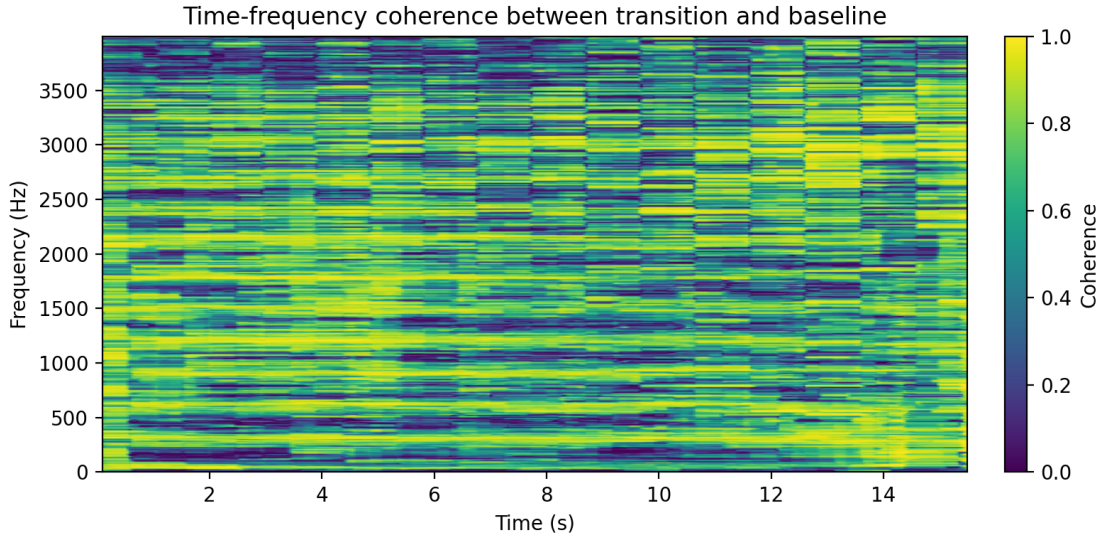

Figure 4.2: Time–frequency coherence between the baseline crossfade and the latent transition, zoomed in to the 0–4000 Hz region.

Looks like a striped pattern in the frequency axis. To find if there are any patterns, and to visualize this more clearly, a mean coherence plot is found below in Figure 4.3

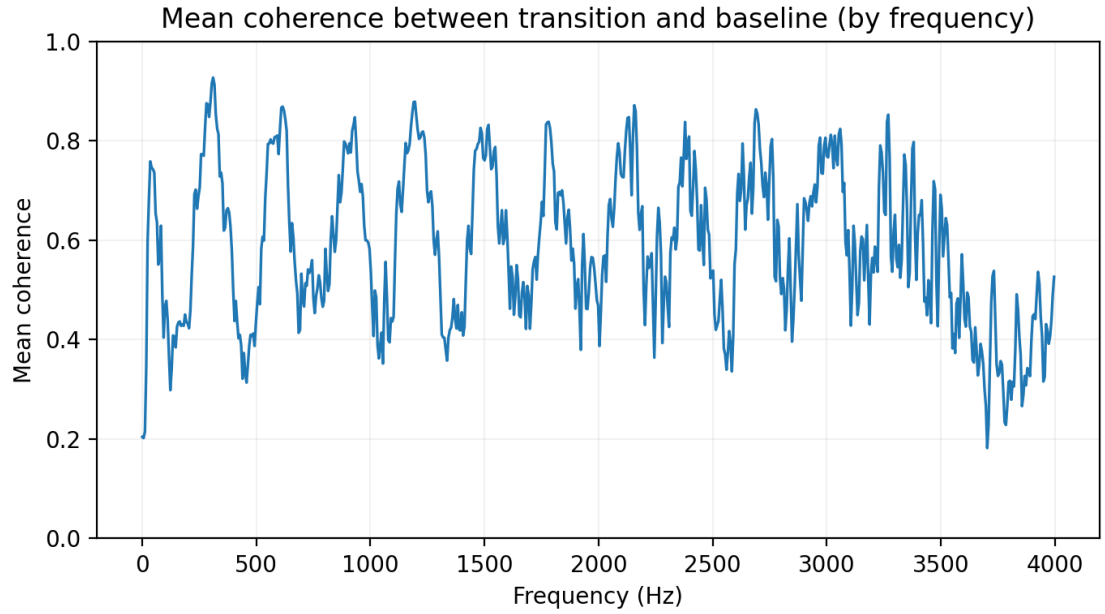

Figure 4.3: Mean coherence over time as a function of frequency, zoomed in to the 0–4000 Hz region.

There seems to be a periodic structure in the mean coherence, so the most prominent peaks are calculated according to Section 3.3.3 and shown in Table 4.3 on the next page.

Rank	Peak (Hz)	Coherence
1	310.5	0.927
2	1195.3	0.878
3	2156.2	0.871
4	615.2	0.869
5	2689.5	0.863
6	3269.5	0.852
7	931.6	0.847
8	1781.2	0.838

Table 4.3: Peak/spacing summary from mean coherence for the representative transition.

From the figures and table above, one can conclude that for this experiment, coherence is highest in the region just above 300 Hz, along with its multiples.

4.1.3 Qualitative audio diagnostics

Spectrogram comparisons

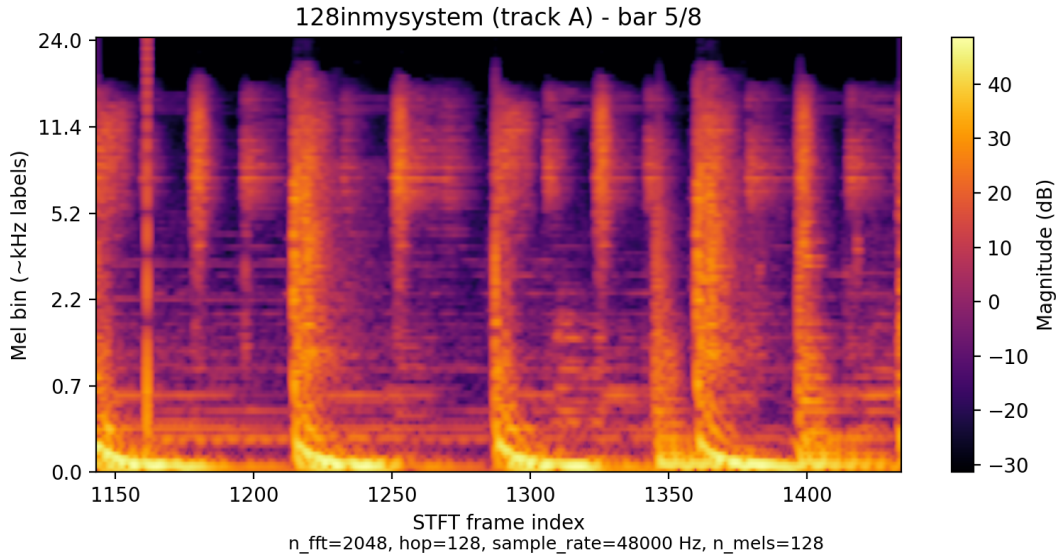

Figure 4.4: Spectrogram of the source track A segment at bar 5.

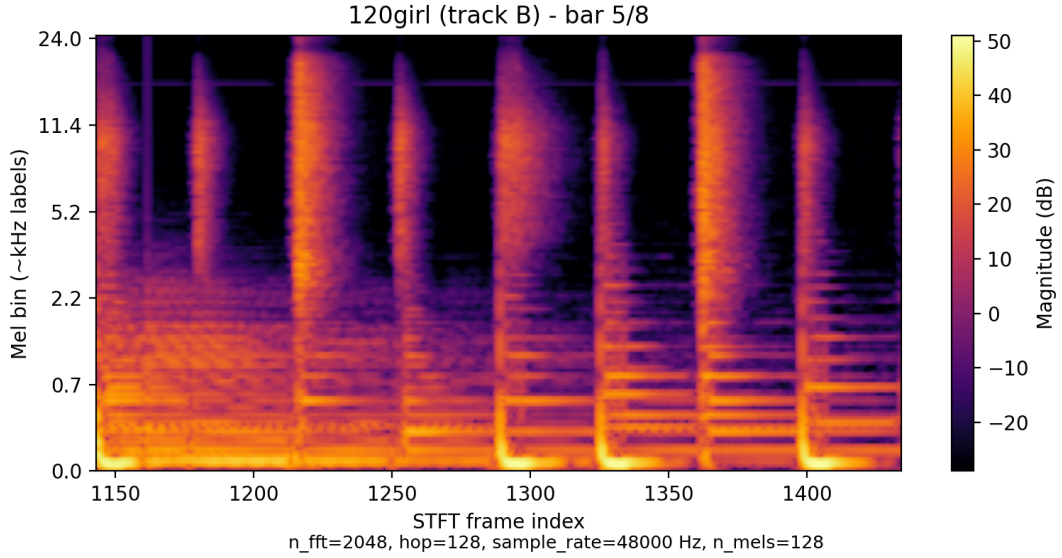

Figure 4.5: Spectrogram of the source track B segment at bar 5.

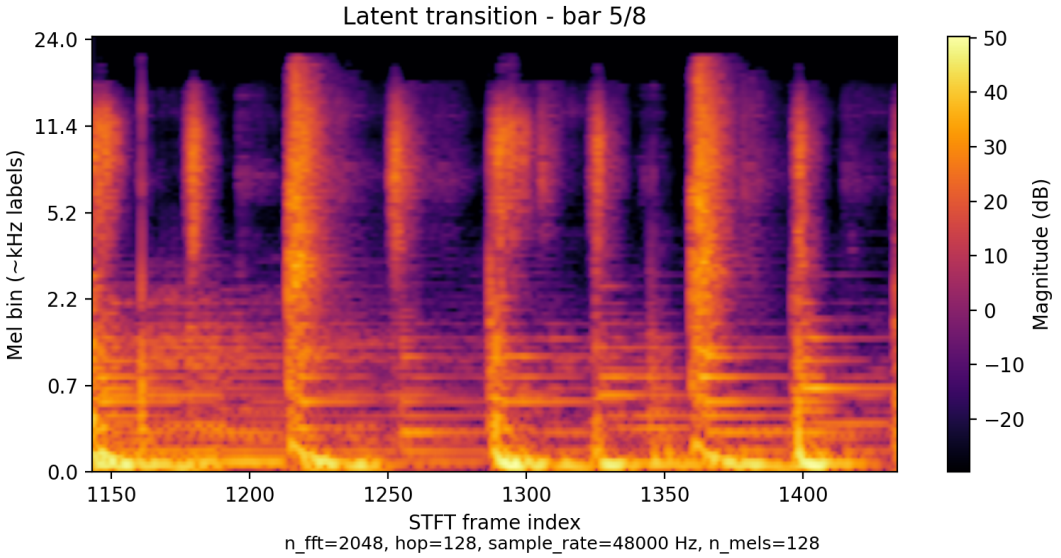

Figure 4.6: Spectrogram of the blended latent transition at bar 5.

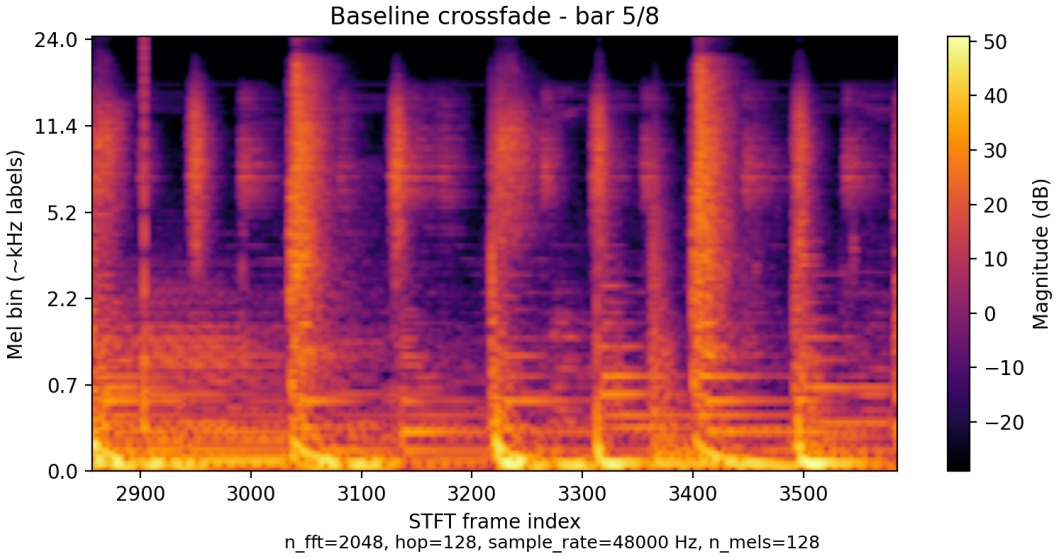

Figure 4.7: Spectrogram of the baseline transition at bar 5.

The spectrograms of the latent interpolation and the baseline transition look very similar, albeit with a somewhat smoother structure and less detail in the latent interpolation. Both tracks' original spectra are also observable in the transition plots.

4.2 Latent and diagnostic analyses

This section summarizes diagnostics in the representation space that complement the objective metrics.

4.2.1 Latent trajectory diagnostics

PCA Trajectories

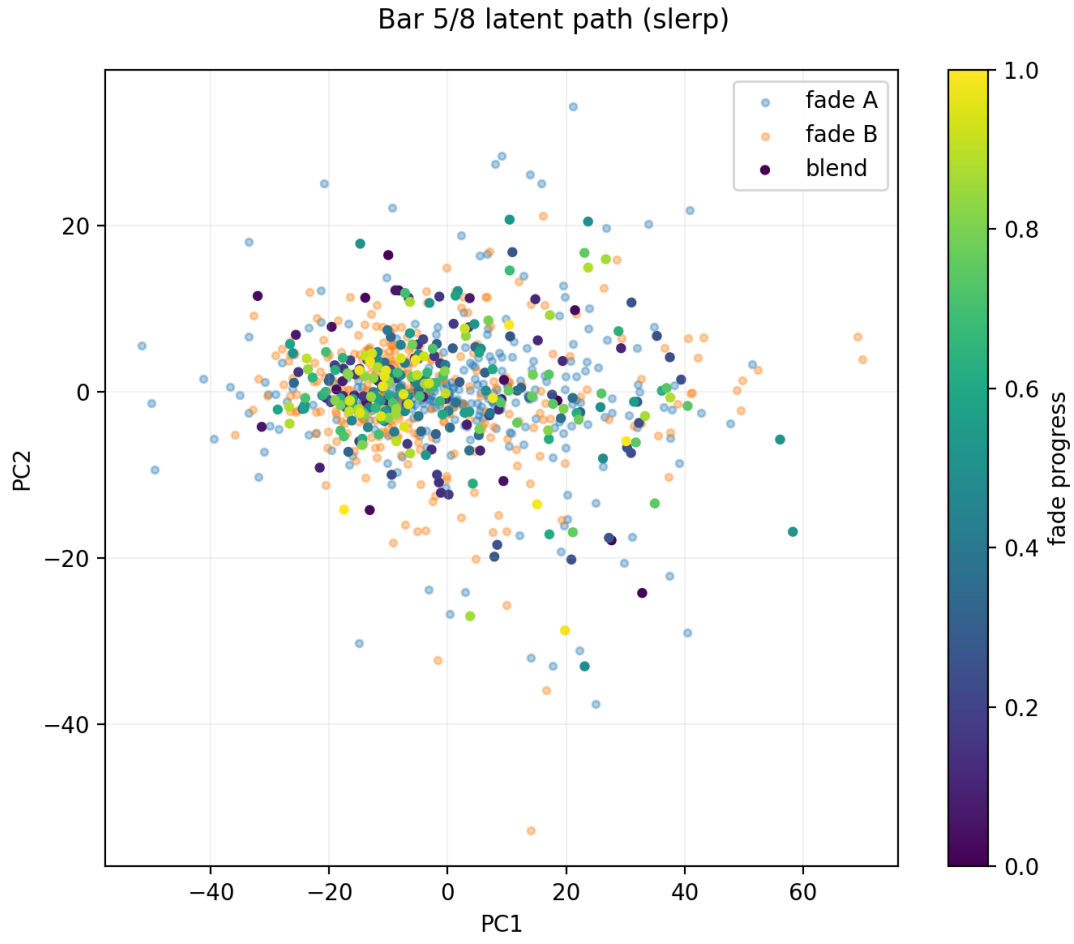

Figure 4.8: PCA trajectories for the representative transition (bar 5 of the fade window). The animation available in the repository is easier to follow, as we have lots of frames (link in Appendix D). As for this plot, one can conclude that the clouds of the different latent sequences overlap a lot.

Component	Explained variance ratio
PC1	0.470572 (47.057%)
PC2	0.084789 (8.479%)
PC3	0.050709 (5.071%)
PC4	0.028017 (2.802%)
PC5	0.020655 (2.065%)

Table 4.4: Top explained variance ratios for the PCA projection. We can conclude that Figure 4.8's two axes explain over **55%** of the variance in latent space.

PCA DTFT

Note that the plot and analysis are evaluated on a single track. We chose track B because it has a BPM of 120, which should have dominant peaks of 2 Hz.

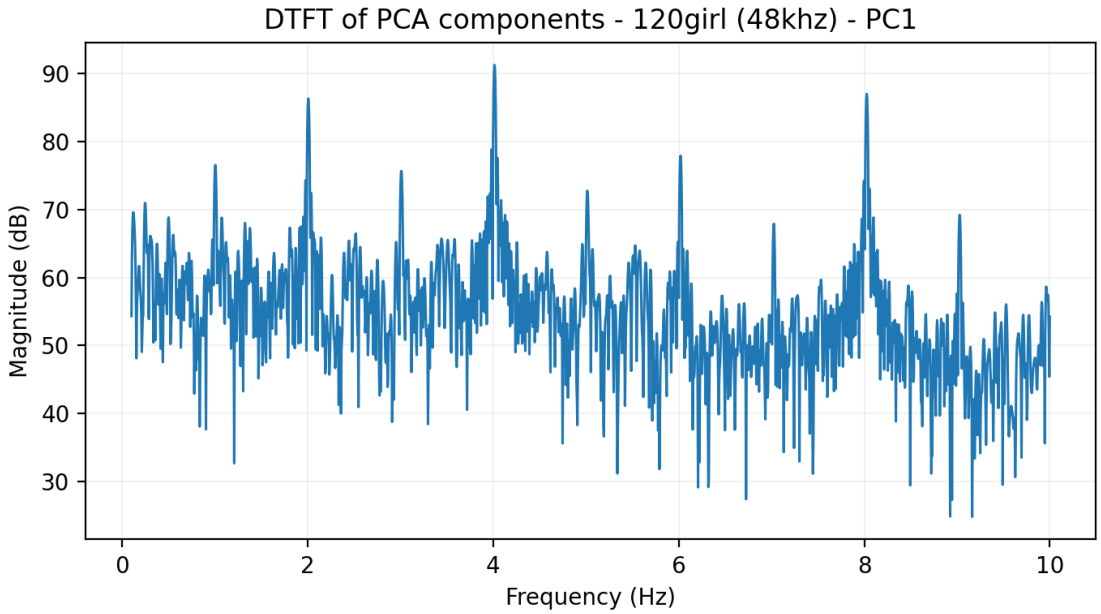

Figure 4.9: DTFT magnitude of the first PCA component for track B. There are dominant peaks at the whole integers, which are accentuated at multiples of 2.

Transition cosine similarity

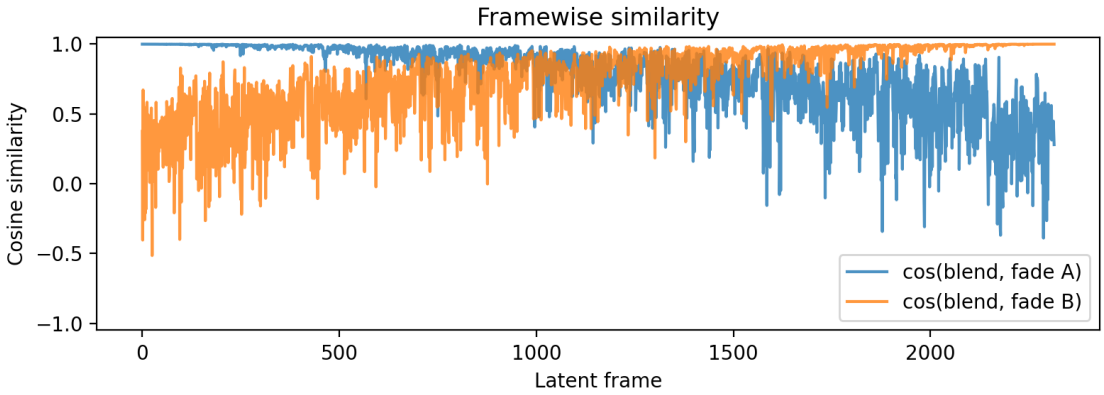

Figure 4.10: Framewise cosine similarity between the blended latent path and each source. Seems to be gradual as expected but not completely smooth, possibly due to transients and percussion material.

4.2.2 Latent structure diagnostics

The correlation matrix and frequency-contribution probe provide complementary views of channel structure: correlated channels suggest shared structure, and spectral loss highlights channels that contribute more strongly to particular frequency regions.

Channel Correlation Structure

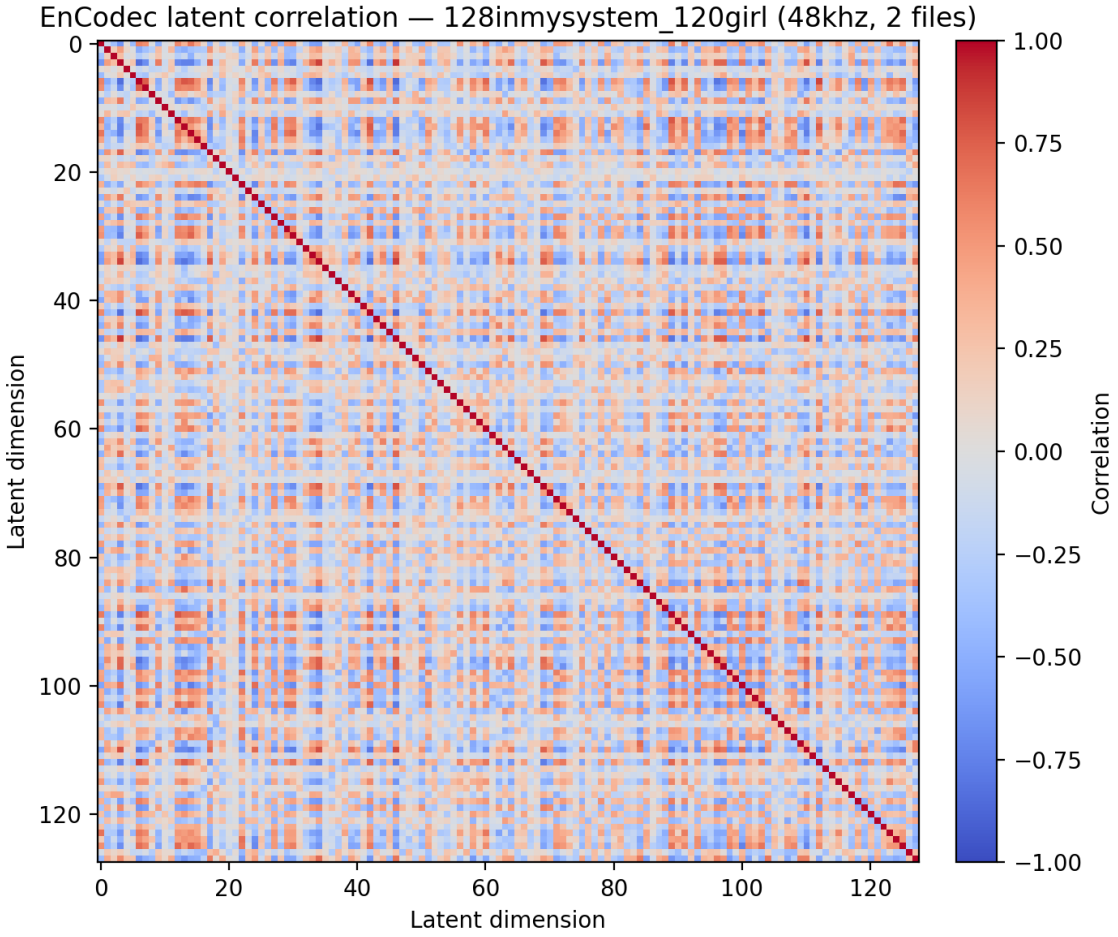

Figure 4.11: Latent channel correlation matrix computed over the representative pair.

Looks like a randomized pattern in the channel correlation matrix in Figure 4.11, which has the correlation computed based on track A and track B, non-transition.

Latent channels frequency contribution

This is meant to show the pattern of contributions from latent channels to specific frequency bands. As EnCodec has 128 latent channels, we are mainly interested in the underlying structure of how different latent channels contribute.

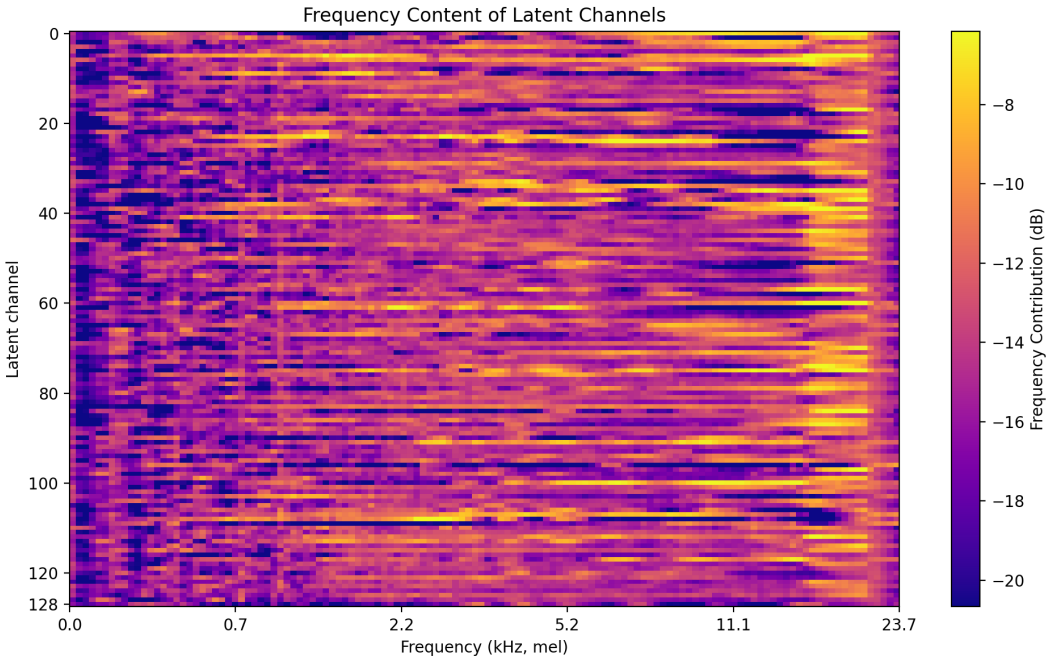

Figure 4.12: Frequency contribution of latent channels shown in a mel-spectrogram, computed by originally taking the spectral loss from zeroing a channel. Brighter areas mean more frequency contribution.

Above, Figure 4.12 gives a clear view that the latent channels differ in frequency contributions, as the plot shows an irregular pattern between the rows. It also shows more discrepancy and relative loss between channels in lower frequency regions. There seems to be no single latent channel that handles a specific frequency band on its own.

4.2.3 Pitch-coding diagnostics

This block follows the pitch-coding diagnostics described in Section 3.4.3. The aim is to isolate how pitch changes (fundamental frequency and overtone structure) are reflected in EnCodec latent space.

Trial	Mean cosine	Min cosine	Max cosine
Tone pair (A4 vs. B4)	0.555	0.273	0.782
Real song (+1 semitone)	0.416	-0.770	0.978

Table 4.5: Cosine similarity summary for the controlled tone-pair trial and the real-song semitone-shift trial.

Controlled harmonic tone-pair trial (A4 vs. B4)

Figure 4.13 projects the latent trajectories for the A4 and B4 tones onto the shared PCA basis. The trajectories form partially separated clouds, indicating that a pitch change alone produces a measurable shift in the latent representation, even with synthesis settings held

fixed (overtone count, rolloff, envelope, and duration). In addition, cosine similarity between the two latent sequences remains moderate on average (Table 4.5), suggesting that the representation is not invariant to pitch.

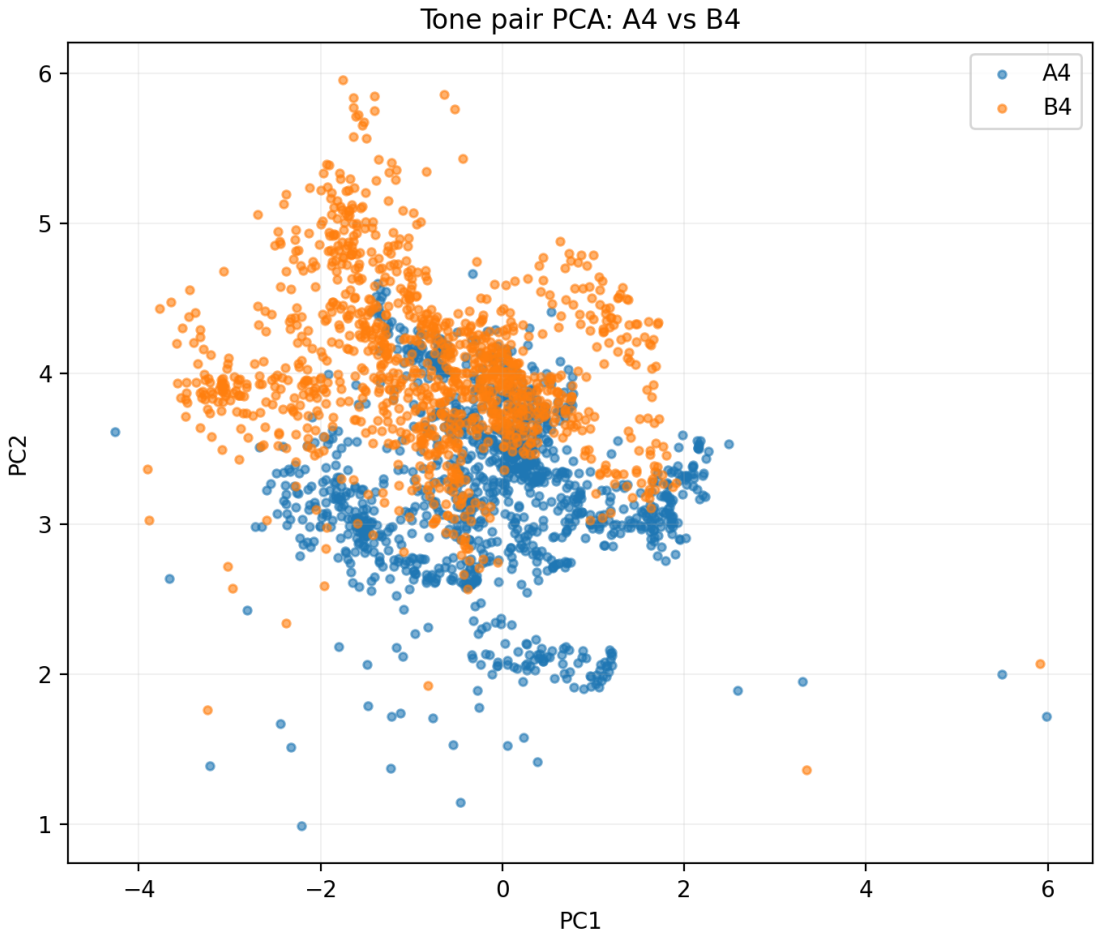

Figure 4.13: Controlled tone pair PCA: A4 (440 Hz) vs. B4 (494 Hz), projected onto the shared PCA basis.

Harmonic sweep trial (A4 → A5)

Figure 4.14 plots cosine similarity between the sweep latent and two references (A4 and A5) using fundamental frequency as the x-axis.

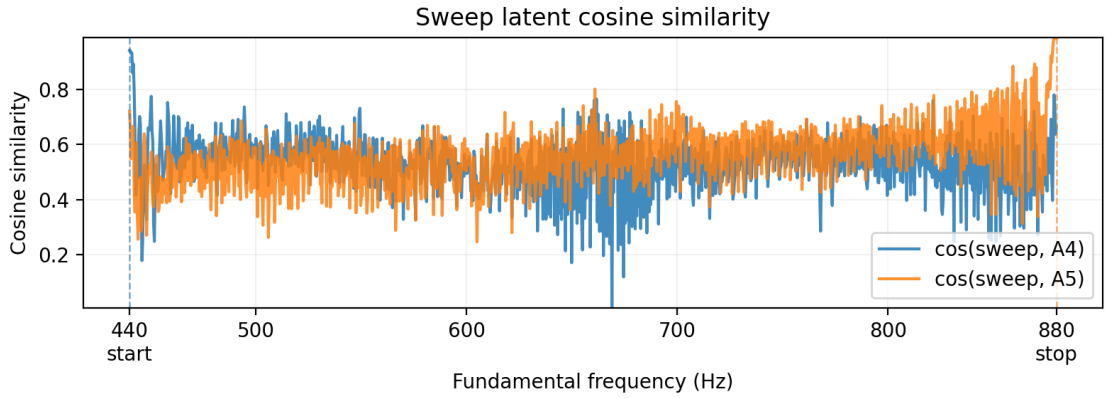

Figure 4.14: Sweep latent cosine similarity vs. fundamental frequency: cosine to start reference (A4) and to end reference (A5).

Similarity to the end reference tends to increase toward the stop frequency, while similarity to the start reference decreases. At the same time, there is still a large jump at the beginning and end, suggesting that the latent space is very sensitive to pitch-change.

Track semitone-shift trial (+1 st)

Figure 4.15 shows frame-wise cosine similarity between the original and one semitone-shifted latents. The wide spread (including negative cosine values) indicates that pitch shifting affects the representation in a content-dependent manner across time (e.g., different notes, transients, and mix states).

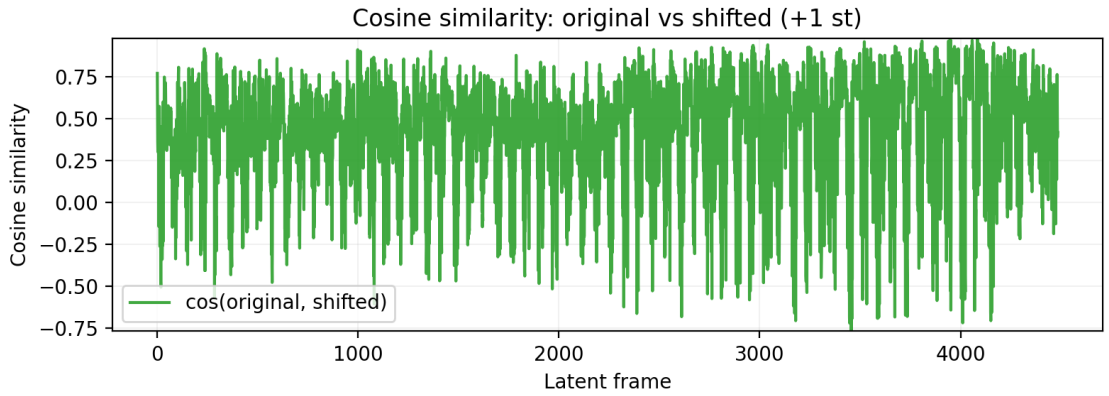

Figure 4.15: Real-song semitone shift: cosine similarity between original and +1 semitone latents over time.

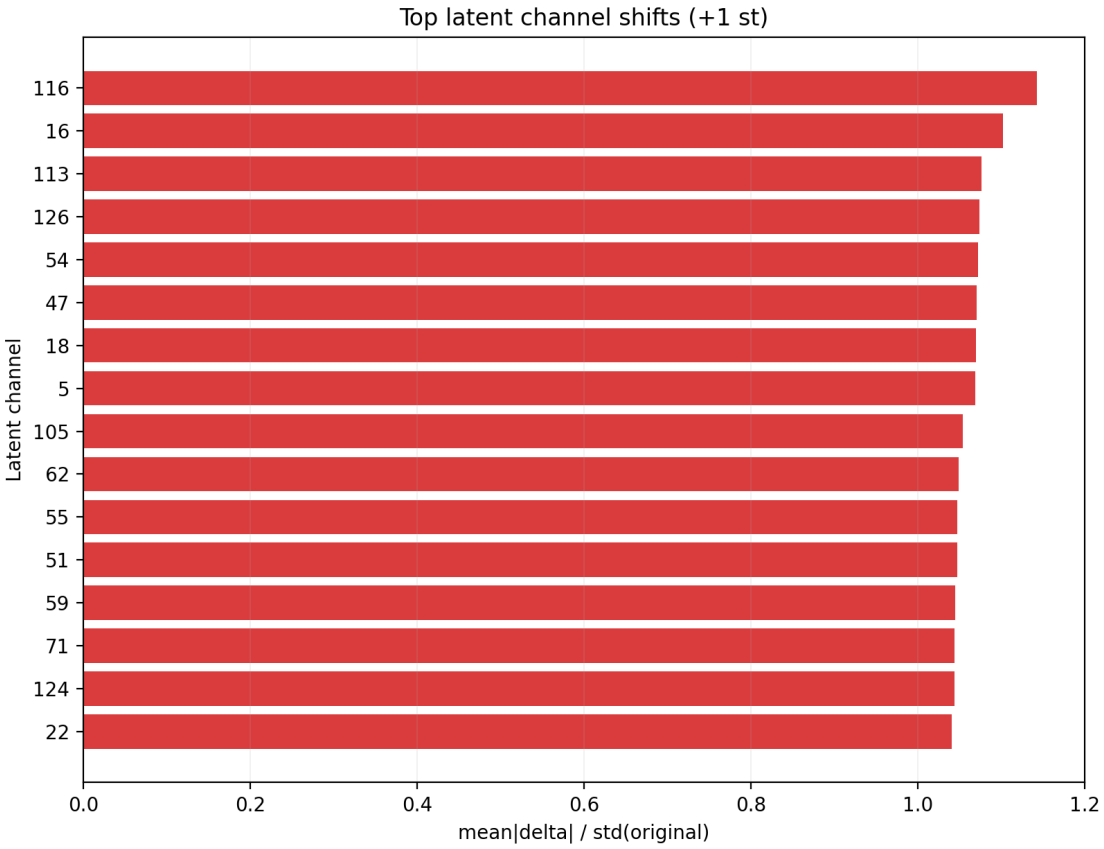

Figure 4.16: Top latent channel shifts for the **+1** semitone song trial, ranked by normalized mean|Δ| / std(original). While channels 116 and 16 are the most shifted channels from the pitch-shift, they are still just slightly above 1 in value.

4.2.4 Latent magnitude vs decoded power

Latent space energy vs. decoded waveform power

The plot below shows the latent space L2 norm in the fade region of both tracks as well as the blended latent series. This is compared to the decoded waveform power to investigate structural correlation between these metrics.

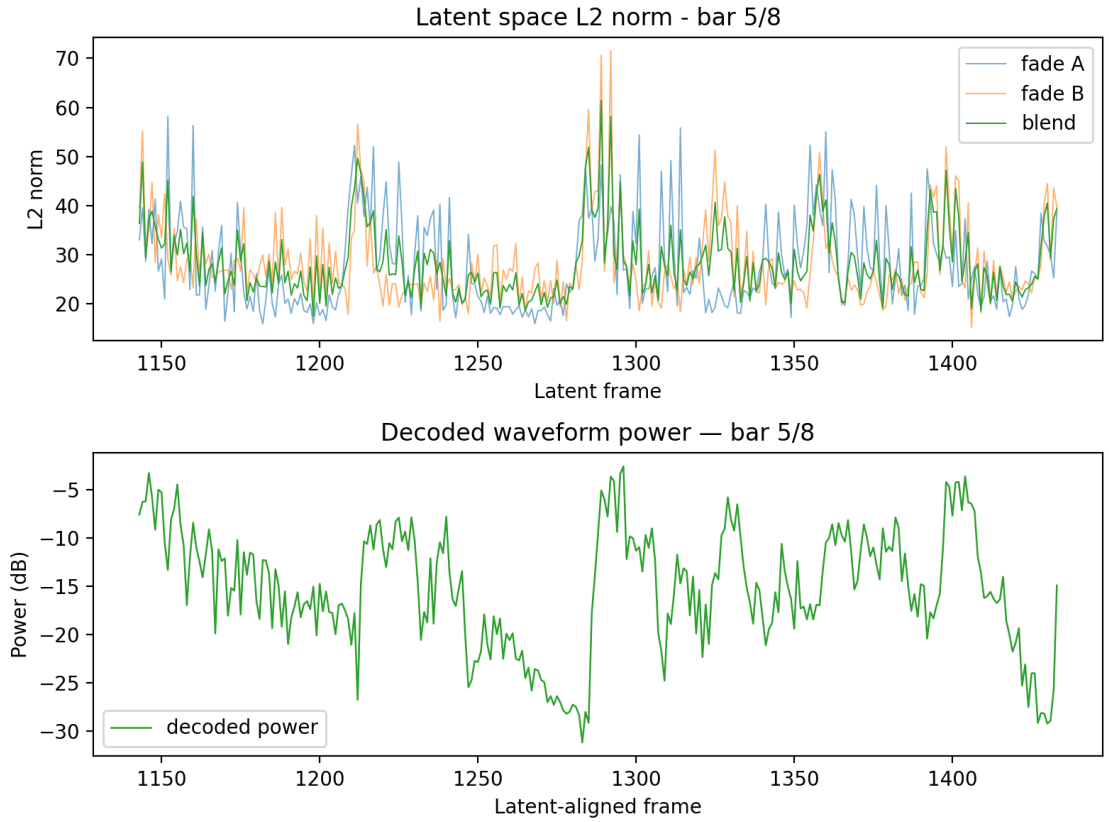

Figure 4.17: Latent L2 norm and waveform power for the latent interpolation transition (bar 5).

Figure 4.17 shows similar structure between the L2 norm of the latents and the waveform power, albeit at what seems to be a small delay. The delay is investigated further below in the cross-correlation plot.

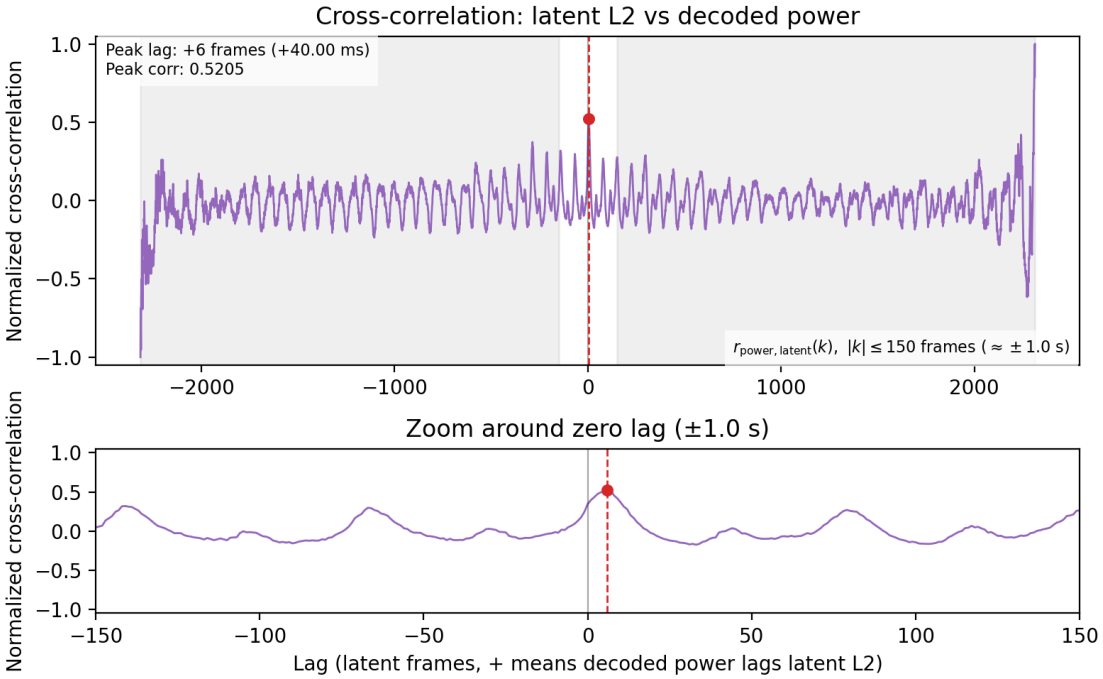

Figure 4.18: Cross-correlation between latent L2 norm and decoded waveform power for the representative transition, shown as $r_{\text{power,latent}}(k)$.

Figure 4.18 confirms that the strongest correspondence occurs close to zero lag, with the peak at a small positive lag. For this representative transition, the peak is at $k^\star = +6$ latent frames (40 ms at 48 kHz), indicating that decoded waveform power follows latent L2 changes with a short delay. Edge cases are irrelevant, as the windows are too small to analyse (goes down to one frame).

Chapter 5
Discussion

This chapter interprets the results in Chapter 4 and discusses what they imply for latent-space music transitions in practice. The goal is to explain why the observed behavior occurs and what it means to use a neural codec latent space as a domain for transitions or other signal processing for music. The discussion is organized to mirror the evaluation, where we first summarize the main findings, then compare latent interpolation to a standard equal-power crossfade, and then use the diagnostic analyses to reason about what happens inside the latent representation. We also discuss practical trade-offs when working with EnCodec, and end with practical implications but also limitations. This discussion provides the context for the conclusions in Chapter 6.

5.1 Summary of Key Results

Across the evaluation in Chapter 4, latent interpolation in the EnCodec representation produces transitions that are comparable to an equal-power crossfade in terms of large-scale spectral structure, but it does not behave exactly like a waveform-domain mix. The overall agreement is strongest in frequency regions that typically carry harmonic and melodic content, while deviations become more obvious in higher-frequency detail, which suggests that the codec representation and decoder are constructed with more emphasis on the musical spectral range.

The latent-space diagnostics indicate that the transition follows the intended fade window, but they also show that the latent representation is not evenly structured in the compressed latent space. Some channels appear to matter more than others for the decoded sound, as they contribute to different spectra. There are also groups of channels that behave similarly. This suggests that blending in latent space is not equivalent to blending independent “stems”, but rather a mix inside a representation where information is spread across dimensions. As a result, it is reasonable that the latent transition can sound different from a crossfade even

when the timing and fade envelope are matched.

The magnitude-based analysis further shows that changes in latent magnitude follow changes in decoded energy, but with a small delay. This is consistent with the codec operating on frames and with smoothing introduced during reconstruction from the overlapping regions. Overall, the results support the view that EnCodec latent interpolation can produce musically coherent, bar-aligned transitions, but that the perceived quality depends on how the codec represents the two sources and on practical choices such as alignment, scale handling, and other preprocessing in the pipeline. It also shows some promise for latent space interpolation as an alternative to the standard equal-power crossfade technique on the waveform for musical transitions.

5.2 Interpreting Transition Quality vs. the Baseline

The equal-power crossfade provides a clear reference point because it is a purely waveform-domain operation where the transition is created by summing two audio signals under a fixed amplitude envelope. This means that the baseline preserves the fine temporal structure of both sources throughout the fade window, and any mismatch is primarily caused by musical incompatibility (e.g., mismatched rhythmic phase, conflicting harmony, or competing arrangement density) rather than by the blending itself. In this sense, the baseline is predictable as it does not introduce a new synthesis stage, and it does not alter the structure of either track's signal beyond the gain scaling applied over time, so the blended spectral content is merely a combination of both tracks.

Latent interpolation differs because it is not a sum of two waveforms but a trajectory through a learned representation followed by re-synthesis. Even when the fade schedule is constructed to imitate a standard crossfade, the decoded output is constrained by what the codec decoder can represent at each intermediate latent state. This introduces a regularization effect, which can be heard and seen in the smoothed spectrogram in Figure 4.6. Intermediate states tend to be smoother and more "averaged" than the corresponding waveform window, which can reduce some forms of clutter that arise when two dense musical mixes overlap. At the same time, this constraint is also the source of characteristic failures, where the transition can lose high-frequency texture, soften transients, or introduce codec-like artifacts if the intermediate representation does not correspond to a realistic audio frame. This appears to be an outlier case, however, and as the EnCodec model is trained with long-range structure in mind from its LSTM loss it keeps this caveat manageable.

Interpreting the objective coherence results therefore requires separating two effects. First, coherence is a similarity measure relative to the crossfade baseline, so it implicitly treats the baseline as the target behavior. When coherence is high, it supports the view that the latent trajectory produces a blend that is close to what a waveform-domain crossfade would generate under the same alignment and fade window. When coherence drops, it does not necessarily imply that the latent transition is perceptually worse, but rather that it departs from the baseline in time–frequency structure. A portion of the mismatch can be explained

by the codec reconstruction itself, with the explanation that the decoder reconstructs a single waveform rather than a two-source mixture. This means small deviations in fine spectral detail (particularly at higher frequencies) will accumulate even if the global spectral envelope and rhythmic structure remain similar. That said, since coherence is irregular between frequencies and the overall mean is less than 0.5, one can clearly draw the conclusion that the transition methods differ. A periodic component is also visible in the coherence diagnostics: Figure 4.3 shows repeating peaks in the mean coherence curve, and Table 4.3 indicates a dominant spacing near 300 Hz with additional lower-amplitude spacings.

From a perceptual perspective, the baseline and the latent method can therefore be seen as optimizing different notions of "smoothness". A crossfade preserves both sources and their microstructure, which could be desirable when the goal is to keep information and detail. But it can also create perceptual problems from waveform inference which itself is highly dependent on phase from the signal. Latent interpolation instead tends to compress this overlap into a single decoded output, which can sound cleaner in the sense that the transition contains fewer simultaneously competing details, but it can also sound less crisp because the decoder smooths rapid changes and does not guarantee that intermediate latent states correspond to a realistic mixture of the two inputs. This is especially relevant around transient-heavy material (e.g., kicks and hi-hats), where waveform superposition produces a straightforward overlap, while latent interpolation may smear or reshape the transient content depending on how those events are represented across latent frames.

A central practical factor is alignment. A crossfade is a simple waveform mix, as it turns track A down, turns track B up, and adds them together. It will "work" even if the tracks are not aligned, but the misalignment becomes very audible during the overlap. Typical examples are double kicks, stuttered hits, or hi-hat patterns that fight each other, because the tracks are not aligned rhythmically.

Latent interpolation is sensitive in another sense because it blends short time-slices in the compressed latent space. In practice, this assumes that the latent frame at time t in track A corresponds to the same musical moment as the latent frame at time t in track B (same beat position within the bar). If that is not true, the interpolation can end up mixing a "kick moment" from one track with an "in-between beats" moment from the other. When the decoder then reconstructs audio from this mixed latent state, it has to produce one plausible waveform, and the result can sound like something entirely different in a short section than what baseline produced. Transients may smear, the groove can lose punch, and rhythmic clarity can drop in a way that is different from the obvious "two beats at once" effect of a crossfade. Vocals might also "disappear" faster due to those latent channels being overpowered, or not produce a significant difference to the rest of the channels for those specific frames.

Finally, the baseline equal-power crossfade provides explicit energy control through its gain schedule, whereas the latent method only inherits energy control indirectly through latent magnitudes, gain scale and decoder dynamics. This difference matters for transitions, as equal-power mixing aims to keep perceived loudness stable, while latent interpolation can deviate from that behavior because the decoder couples amplitude, timbre, and reconstruction smoothing. As a result, even when the fade envelope is well-defined in latent space, the

energy of the decoded result can shift slightly relative to the baseline, and adjustments such as scale handling and normalization become part of the quality trade-off.

Overall, the comparison suggests that latent interpolation should not be evaluated as a direct attempt to replicate a crossfade, but as an alternative transition operator with different strengths. The crossfade remains the more transparent and robust choice when the objective is to preserve both sources faithfully, while latent interpolation can provide a more perceptually satisfying alternative when audio smoothing and minimizing fine details between complex tracks is desirable.

5.3 What the Diagnostics Suggest About EnCodec Latents

The diagnostics in Section 4.2 are most useful when read as one connected picture of what the latent transition is doing, rather than as isolated figures. Since the plots are computed on the continuous latent vectors that are actually interpolated (i.e., the representation the decoder conditions on), they describe how the transition evolves inside the codec representation before the final waveform is reconstructed. This matters because transition quality is shaped not only by the interpolation schedule, but also by temporal structure, channel coupling, and how the decoder maps intermediate latent states back to audio (Défossez et al., 2022).

The PCA trajectory plot in Figure 4.8 should mainly be read as a qualitative visualization of how irregular latent motion is over time, not as evidence that the interpolation moves cleanly between two well-separated "regions" in latent space. In the static projection, points from the fade window are highly scattered and overlapping, and the blend does not trace a simple path that can be interpreted as traveling from an "A cluster" to a "B cluster." What stands out instead appears in the animation, where motion is uneven and organized in time intervals, as short stretches with relatively small coordinate changes are followed by stretches with larger jumps. This supports an interpretation where PCA is used to indicate when the representation changes quickly versus slowly, without attaching strong musical meaning to the PC1/PC2 directions themselves.

It is also worth noting that the PCA trajectories in Figure 4.8 are a 2D projection of a high-dimensional latent trajectory, and therefore only reflect the fraction of latent variation captured by the plotted subspace. For the representative transition, Table 4.4 shows that PC1 and PC2 together explain a bit over 55% of the latent variance, meaning that a substantial remainder is distributed across higher components that are not visible in the plot.

The DTFT view in Figure 4.9 is a more directly interpretable complement to the PCA projection because it tests whether the dominant PCA component contains periodic structure at musically relevant rates. For the selected song example (120 BPM), spectral energy is concentrated around the expected beat-rate region at about 2 Hz and its multiples. This suggests that at least one strong latent direction carries tempo-linked repetition. Since beat-synchronous structure in dance music is strongly tied to percussive events, this can reasonably be inter-

preted as rhythmic/percussive content contributing heavily to that component. However, the result should be read as a dominant contribution pattern, not as evidence of a single isolated "percussion-only" axis in latent space.

The cosine similarity curves in Figure 4.10 provide the clearest summary of the handover dynamics. Across the fade window, the blend becomes steadily less aligned with source A and more aligned with source B, which matches the intended gradual transition behavior of a slerp. The smaller local dips are best understood as short moments where the blended latent direction briefly deviates from both sources, rather than as one track "overpowering" the other in amplitude. Since cosine similarity is direction-based (and normalized by the vector norms), these deviations are more naturally explained by transient or frame-local changes in the representation that momentarily produce a latent vector that is not well-aligned with either endpoint, even though the overall handover trend remains smooth.

The channel-level diagnostics explain why these short deviations and the broader "smoothing" character can coexist. The correlation structure (Figure 4.11) indicates that channels are not independent and that subsets move together, while the frequency-contribution probe (Figure 4.12) shows uneven and overlapping influence across the spectrum rather than a clean band-wise separation. Taken together, this points to an entangled representation where multiple channels jointly support the same spectral regions and where changes in a subset of channels can have distributed effects in the decoded audio. This helps explain why interpolation can preserve broad musical structure while still reshaping fine detail, especially in transient-heavy or high-frequency content where small representational changes can alter the reconstructed texture.

The pitch-coding diagnostics in Section 4.2.3 add a complementary constraint by isolating how pitch changes are reflected in EnCodec latent space. Even with synthesis settings held fixed, the controlled tone-pair trial (A4 vs. B4) produces only moderate cosine similarity on average (Table 4.5) and partially separated PCA trajectory clouds (Figure 4.14), which indicates that the representation is not pitch-invariant. The harmonic sweep (A4→A5) further suggests high sensitivity around pitch-change boundaries, as similarity to the start and end references shows large endpoint jumps (Figure 4.15). In real musical material, this effect becomes strongly content-dependent over time: a +1 semitone shift yields a wide spread in frame-wise cosine similarity (including negative values; Table 4.5 and Figure 4.16), consistent with pitch interacting with note content, transients, and mix state rather than acting as a uniform global offset. Finally, the per-channel shift ranking (Figure 4.17) suggests that pitch-related changes are not cleanly localized to a single "pitch channel"; even the most affected channels remain only modestly elevated under the normalized change metric. Taken together, these trials imply that pitch/key mismatch between sources can move the latent trajectory through substantially different regions, which can plausibly increase the risk that intermediate interpolants fall outside what the decoder reconstructs cleanly.

Finally, the L2 norm diagnostic in Figure 4.17 links latent dynamics to decoded energy across the transition. The latent magnitude shows structured rises on the downbeats followed by drops that co-vary with the decoded waveform-power trend, and the plot indicates a clear amplitude relationship where higher latent magnitude generally corresponds to higher de-

coded signal level. Peaks in latent magnitude and decoded power are not perfectly aligned in time, which motivated the further cross-correlation analysis. Figure 4.18 makes this delay explicit by showing that the cross-correlation between the latent L2 norm and decoded waveform power peaks at a small positive lag of $k^\star = +6$ latent frames (40 ms at 48 kHz). This supports the interpretation that changes in latent magnitude slightly precede changes in decoded waveform power, which is consistent with frame-based processing and smoothing across neighboring frames (Défossez et al., 2022; Oppenheim and Schafer, 2010). The offset is short relative to bar-scale timing, so it does not affect the macro transition structure, but it helps explain the local peak misalignment observed in Figure 4.17.

5.4 Practical Considerations and Limitations

A key practical consideration is that EnCodec reconstruction is frame-based and implemented with chunking and overlap-add (Défossez et al., 2022; Oppenheim and Schafer, 2010). This generally helps avoid boundary artifacts, but it also means the latent transition is mediated by a re-synthesis stage that can smooth fast details compared to direct waveform mixing. This is most noticeable in transient-heavy regions, where small differences in latent state from chunk edge processing can change the reconstructed sharpness.

Bandwidth and quantization choices further affect the transition. Higher bandwidths preserve more detail, while lower bandwidths tend to simplify texture and reduce high-frequency content. Optional RVQ re-quantization is also a trade-off, as snapping interpolated latents back to codebook entries can make the representation more codec-consistent, but it may introduce small discontinuities if the quantized trajectory changes abruptly between frames.

The pipeline is also sensitive to preprocessing. Beat and bar alignment is important because interpolation assumes that corresponding latent frames represent comparable musical time. Misalignment, BPM estimation errors, or time-stretch artifacts can therefore reduce rhythmic clarity in the fade region, even when the overall envelope is correct. Pitch/key mismatch is another sensitivity: the pitch-coding diagnostics in Section 4.2.3 show that even controlled pitch changes can shift the latent representation substantially (Table 4.5), so transitions between harmonically distant sources (or preprocessing that introduces pitch drift) may require explicit key alignment to avoid traversing unrealistic intermediate latents. Scale handling and normalization matter for the same reason: loudness through the transition is not only determined by the fade curve, but also by how the codec maps intermediate latent magnitudes and scales back to waveform amplitude.

In terms of limitations, the experiments focus on a small set of bar-aligned transitions in steady-tempo electronic music, so the conclusions may not transfer directly to material with tempo drift, irregular phrasing, or very sparse and irregular arrangements. For those types of transitions, beatmatching is not a viable option and a DJ would have to rely on effects or other alternatives.

Chapter 6
Conclusions

6.1 Main Conclusions

This thesis evaluated whether bar-aligned latent-space interpolation using a pretrained neural audio codec (EnCodec) can produce track-to-track transitions that outperform a conventional equal-power waveform crossfade. Across the AB listening evaluation, latent interpolation (SLERP) was preferred at a similar rate as the baseline, with a substantial fraction of trials marked as "no preference", indicating comparable perceptual quality rather than a clear winner. Objective similarity diagnostics further support that the latent method preserves the broad time–frequency structure of the transition, but it does *not* behave like waveform-domain mixing and tends to smooth fine detail, particularly in transient- and texture-heavy regions. Analyses in representation space suggest that the codec latents have meaningful structure (e.g., tempo-linked periodicity and pitch sensitivity), but that this structure is distributed and entangled across channels rather than cleanly factorized. Overall, latent interpolation should be understood as a different transition operator than crossfading, meaning it is not inherently better or worse in quality. Instead, it offers distinct behavior and new opportunities because the transition is performed in a highly compressed representation.

6.2 Practical Implications

For practical automated transitions, the strongest takeaway is that latent interpolation can be used as a *viable alternative* to a crossfade when the goal is a coherent, bar-aligned handover, but it should not be treated as a replacement of DJ's. A crossfade is transparent and robust in the way that it preserves the microstructure of both sources which also can be its caveat mainly due to musical incompatibility (timing conflicts, key clashes, arrangement density). Latent interpolation, in contrast, constructs an intermediate latent trajectory and then resynthesizes a *single* waveform. In favorable cases it reduces the perception of "two dense mixes

on top of each other" and yields a smoother, more unified transition acting as a source of regularization. In unfavorable cases it can soften transients, reduce high-frequency texture, distort vocals or introduce codec-like artifacts when intermediate states are difficult to decode cleanly.

The results also imply that preprocessing decisions are first-order determinants of quality. While both approaches benefit from correct beat and bar alignment, latent interpolation is especially sensitive to misalignment because it assumes that corresponding latent frames represent comparable musical moments (e.g., the same beat phase within the bar). If this fails, interpolation may blend mismatched events (kick vs. off-beat space, transient vs. sustain), forcing the decoder to produce a plausible waveform from an implausible intermediate representation, resulting in something that does not sound like either sources. The pitch-coding diagnostics indicate that EnCodec latents change when pitch changes, i.e., the representation is not pitch-invariant. As a result, latent-space operations such as interpolation will inherently mix pitch-dependent representations, but this thesis did not evaluate how pitch/key distance affects transition preference or artifact rates.

Finally, the latent approach introduces a practical trade-off between controllability and complexity. Waveform crossfades offer explicit energy control through the gain curves. Latent interpolation inherits loudness behavior through latent magnitudes, any carried scale envelopes, and decoder dynamics, which can produce some deviations from equal-power behavior. This makes normalization, scale handling, and consistent fade window selection more important than in the baseline. In exchange, operating in latent space opens the possibility of performing other "mix-like" manipulations in a compressed domain, which may be attractive when computational constraints matter or when a more "morphed" transition aesthetic is desired.

6.3 Future Work

The experiments in this thesis represent a deliberately constrained first evaluation using a fixed codec, a small set of transitions, and interpolation as the primary latent operation. The following next steps are concrete directions that would materially strengthen both the empirical conclusions and the applicability of latent-domain transitions:

- **Scale up and broaden the evaluation protocol.** Increase the number of transition pairs and participants, include a wider range of genres (including material with less rigid phrasing), and report effect sizes and confidence intervals in addition to preference counts. In the same evaluation, vary fade lengths and musical conditions (key distance, arrangement density, transient density) to map when latent interpolation is most likely to match or surpass crossfading.

- **Learned refinement of the interpolated segment.** Treat interpolation as a *proposal* and add a final generative refinement stage. One option is a diffusion model (waveform- or latent-domain) conditioned on the tail of track A and the head of track B, tasked with denoising or enhancing the interpolated region while preserving bar alignment.

A related variant is to inject noise specifically into the fade region and let a conditional diffusion model synthesize a more "intentional" bridge than a purely geometric interpolation, potentially improving realism and reducing codec artifacts.

- **Direct latent space editing beyond interpolation.** The diagnostics suggest that EnCodec latents reliably capture musically meaningful patterns such as rhythmic periodicity and pitch related structure, even if these patterns are not cleanly separated into independent latent channels. A practical next step is to train small control models such as linear probes or small MLPs that map simple latent statistics to audible qualities like low versus high frequency balance, transient strength, or how bright it sounds. You can then apply EQ like or dynamics like changes directly in latent space. This would test the thesis claim that working in a compressed representation can make some effects cheaper than doing them on the raw waveform.

- **Automated transition point selection and alignment.** Extend preprocessing from manual bar selection to automatic discovery of good entry/exit regions using novelty curves, structural segmentation, beat/downbeat confidence, and energy profiling. This could automate start/stop bar choices and fade length, and it would make latent interpolation more deployable by reducing the amount of user intervention required for reliable alignment.

- **Use codec latents for retrieval and classification tasks.** Since the representation is compact and appears to preserve musically meaningful structure, it is promising as an embedding for track similarity, clustering, or transition recommendation (e.g., grouping by rhythmic feel, instrumentation density, or timbral profile). Evaluating classification and retrieval performance using latent embeddings would test whether the same compressed space that supports interpolation can also reduce computational costs for tasks such as sample grouping, category labeling, or candidate-pair selection in an automated mixing pipeline.

References

Agostinelli, A., Denk, T. I., Borsos, Z., Engel, J., Verzetti, M., Caillon, A., Huang, Q., Jansen, A., Roberts, A., Tagliasacchi, M., Sharifi, M., Zeghidour, N., and Frank, C. (2023). Musiclm: Generating music from text.

Allen, J. B. and Rabiner, L. R. (1977). A unified approach to short-time fourier analysis and synthesis. *Proceedings of the IEEE*, 65(11):1558–1564.

Argüello, G., Lanzendörfer, L. A., and Wattenhofer, R. (2024). Cue point estimation using object detection.

Bittner, R. M., Gu, M., Hernandez, G., Humphrey, E. J., Jehan, T., McCurry, H., and Montecchio, N. (2017). Automatic playlist sequencing and transitions. In Cunningham, S. J., Duan, Z., Hu, X., and Turnbull, D., editors, *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017), Suzhou, China, October 23–27, 2017*, pages 442–448.

Borsos, Z., Marinier, R., Vincent, D., Kharitonov, E., Pietquin, O., Sharifi, M., Roblek, D., Teboul, O., Grangier, D., Tagliasacchi, M., and Zeghidour, N. (2022). Audiolm: a language modeling approach to audio generation.

Butler, M. J. (2006). *Unlocking the Groove: Rhythm, Meter, and Musical Design in Electronic Dance Music*. Indiana University Press.

Chen, B.-Y., Hsu, W.-H., Liao, W.-H., Martínez Ramírez, M. A., Mitsufuji, Y., and Yang, Y.-H. (2022). Automatic dj transitions with differentiable audio effects and generative adversarial networks. ICASSP 2022; arXiv:2110.06525.

Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., and Défossez, A. (2023). Simple and controllable music generation. NeurIPS 2023.

Défossez, A., Zeghidour, N., Usunier, N., Delétang, G., Synnaeve, G., and Adi, Y. (2022). High fidelity neural audio compression. *arXiv*.

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., and Sutskever, I. (2020). Jukebox: A generative model for music.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*.

Gritsenko, A. A., Salimans, T., van den Berg, R., Snoek, J., and Kalchbrenner, N. (2020). A spectral energy distance for parallel speech synthesis.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Huron, D. (2006). *Sweet Anticipation: Music and the Psychology of Expectation*. MIT Press.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. CVPR 2019.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. (2021). Diffwave: A versatile diffusion model for audio synthesis. In *9th International Conference on Learning Representations (ICLR 2021)*. OpenReview.net.

Krumhansl, C. L. (1990). *Cognitive Foundations of Musical Pitch*. Oxford University Press.

Laroche, J. and Dolson, M. (1999). Improved phase vocoder time-scale modification of audio. *IEEE Transactions on Speech and Audio Processing*, 7(3):323–332.

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

Lerdahl, F. and Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. MIT Press.

Liu, H., Chen, Z., Yuan, Y., Mei, X., Liu, X., Mandic, D., Wang, W., and Plumbley, M. D. (2023). Audioldm: Text-to-audio generation with latent diffusion models. ICML 2023.

Müller, M. (2015). *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer.

Oppenheim, A. V. and Schafer, R. W. (2010). *Discrete-Time Signal Processing*. Pearson, 3 edition.

Plomp, R. and Levelt, W. J. M. (1965). Tonal consonance and critical bandwidth. *The Journal of the Acoustical Society of America*, 38(4):548–560.

Pohlmann, K. C. (2010). *Principles of Digital Audio*. McGraw-Hill, 6 edition.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks.

Roberts, A., Engel, J., Raffel, C., Hawthorne, C., and Eck, D. (2018). A hierarchical latent vector model for learning long-term structure in music. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4364–4373. PMLR.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Russell, S. and Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson, 4 edition.

Schneider, F., Kamal, O., Jin, Z., and Schölkopf, B. (2023). Moûsai: Text-to-music generation with long-context latent diffusion.

Sethares, W. A. (2005). *Tuning, Timbre, Spectrum, Scale*. Springer, 2 edition.

Slaney, M. (1998). Auditory toolbox. Technical Report 1998-010, Interval Research Corporation.

Stevens, S. S., Volkmann, J., and Newman, E. B. (1937). A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190.

Tatar, K., Cotton, K., and Bisig, D. (2023). Sound design strategies for latent audio space explorations using deep learning architectures.

Temperley, D. (2001). *The Cognition of Basic Musical Structures*. MIT Press.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. (2017a). Neural discrete representation learning.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. (2017b). Neural discrete representation learning. *arXiv*.

Yamamoto, R., Song, E., and Kim, J.-M. (2020). Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram.

Yang, D., Xie, Y., Yin, Y., Wang, Z., Yi, X., Zhu, G., Weng, X., Xiong, Z., Ma, Y., Cong, D., Liu, J., Huang, Z., Ru, J., Huang, R., Wan, H., Wang, P., Yu, K., Wang, H., Liang, L., Zhuang, X., Wang, Y., Guo, H., Cao, J., Ju, Z., Liu, S., Cao, Y., Weng, H., and Zou, Y. (2026). Heartmula: A family of open sourced music foundation models.

Zeghidour, N., Luebs, A., Omran, A., Skoglund, J., and Tagliasacchi, M. (2021a). Soundstream: An end-to-end neural audio codec.

Zeghidour, N., Luebs, A., Omran, A., Skoglund, J., and Tagliasacchi, M. (2021b). Soundstream: An end-to-end neural audio codec.

Zehren, M., Alunno, M., and Bientinesi, P. (2020). Automatic detection of cue points for dj mixing.

Zölzer, U., editor (2011). *DAFX: Digital Audio Effects*. John Wiley & Sons, 2 edition.

Appendices

Appendix A
Abbreviations

AAC . . . . . Advanced Audio Coding

AB . . . . . A/B Listening Test

AI . . . . . Artificial Intelligence

BPM . . . . . Beats Per Minute

CLI . . . . . Command-Line Interface

CPU . . . . . Central Processing Unit

DJ . . . . . Disc Jockey

DSP . . . . . Digital Signal Processing

DTFT . . . . . Discrete-Time Fourier Transform

GPU . . . . . Graphics Processing Unit

LSTM . . . . . Long Short-Term Memory

MLP . . . . . Multi-Layer Perceptron

MP3 . . . . . MPEG-1 Audio Layer III

MS-STFT . . . . . Multi-Scale Short-Time Fourier Transform

OLA . . . . . Overlap-Add

PCA . . . . . Principal Component Analysis

PCM . . . . . Pulse Code Modulation

RMS . . . . . Root Mean Square

RNN . . . . . Recurrent Neural Network

RVQ . . . . . Residual Vector Quantization

SLERP . . . . . Spherical Linear Interpolation

STFT . . . . . Short-Time Fourier Transform

VQ . . . . . Vector Quantization

VQ-VAE . . . . . Vector-Quantized Variational Autoencoder

WAV . . . . . Waveform Audio File Format

Appendix B
Supplementary Neural-Network Background

Neurons and Layers

A neuron applies a weighted sum followed by a nonlinearity. For an input vector $\mathbf{x}$, weights $\mathbf{w}$, and bias b, the output is

$$y = \sigma(\mathbf{w}^\top \mathbf{x} + b), \tag{B.1}$$

where $\sigma(\cdot)$ is an activation function such as ReLU or tanh. A full layer stacks many neurons:

$$\mathbf{h} = \sigma(W\mathbf{x} + \mathbf{b}). \tag{B.2}$$

Stacking layers yields a feed-forward network,

$$\mathbf{h}^0 = \mathbf{x}, \qquad \mathbf{h}^l = \sigma\left(W^l \mathbf{h}^{l-1} + \mathbf{b}^l\right), \quad l = 1, \ldots, L, \tag{B.3}$$

with output $\mathbf{h}^L$. The nonlinearity is a dealbreaker since without it, multiple layers collapse to a single linear map. Depth lets the model build progressively richer representations, which is why deep networks can capture complex data patterns (Goodfellow et al., 2016).

Backpropagation

To minimize the supervised objective such as in Eq. 2.2, we need gradients of its loss with respect to all parameters, which backpropagation computes efficiently. Training minimizes a loss $\mathcal{L}(\theta)$ using gradient-based optimization. Backpropagation computes $\nabla_\theta \mathcal{L}$ by applying the chain rule from the output layer back to the input (Rumelhart et al., 1986; Goodfellow et al., 2016). Parameters are updated with gradient descent,

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}, \tag{B.4}$$

where η is the learning rate. In practice, gradients are estimated on mini-batches, and the update is repeated until validation performance stops improving. To state the chain rule, the

linear input to a layer (the pre-activation) is defined as $\mathbf{z}^l = W^l \mathbf{h}^{l-1} + \mathbf{b}^l$ and the activated output as $\mathbf{h}^l = \sigma(\mathbf{z}^l)$. The backpropagated error then satisfies

$$\delta^l = (W^{l+1})^\top \delta^{l+1} \odot \sigma'(\mathbf{z}^l), \tag{B.5}$$

which yields the parameter gradients $\nabla_{W^l} \mathcal{L} = \delta^l (\mathbf{h}^{l-1})^\top$ and $\nabla_{\mathbf{b}^l} \mathcal{L} = \delta^l$.

Recurrent Layers (LSTM)

LSTMs add a separate cell state and three gates (input, forget, output) that are computed from the current input and previous hidden state. The gates are sigmoid-valued weights in $[0, 1]$ that decide what to keep, write, and expose. A compact LSTM update is

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \tag{B.6}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{B.7}$$

where $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$ are the gates and $\tilde{\mathbf{c}}_t$ is the candidate update (Hochreiter and Schmidhuber, 1997). This design lets information persist over long ranges while still adapting to new input.

Appendix C
Declaration of AI usage in this thesis

Naturally, neural networks are not only used as the transition operator in the experiments, but have also been a valuable tool in this thesis. I have used OpenAI's Codex to help me test and write small code snippets. However, as a computer science engineer, I am keen on keeping my code readable and organized. Especially the latent pipeline for this project. This means I designed the main pipeline entirely myself, only relying on LLM assistance as a form of pair-programming support. The repository structure, code design, and overall organization were all done by hand. The ideas, discussion, and conclusions are, of course, my own. While LLMs can be helpful, they remain limited when reasoning about new and unexplored problems (for example, latent space interpolation in neural audio codecs..). That said, in practical terms, I would likely have spent much more time on syntax errors and minor issues without Codex.

Language and spelling errors in the thesis have also been mitigated using LLMs. My workflow sometimes involved providing an idea or rough draft and asking an LLM to generate an initial short text stub, mainly to help organize my thoughts and get started. In later stages, I also used LLMs for general spell correction, minor structural improvements, and locating notes I had written but forgotten. I believe the current state of AI is an extremely powerful tool that can be a fantastic assistant when used correctly, but it can also become a way to waste time and learn absolutely nothing if used improperly, rendering useless as an educational tool.

Appendix D
Repository

The code for this thesis is publicly available at:

`https://github.com/olleryden/Examensarbete`

The experiments reported in this thesis were run from this repository. For reproducibility, the thesis version can be pinned to commit `8710590dae9b` (accessed March 3, 2026).

Repository Overview

The repository is organized into the following top-level folders:

- `data/`: input audio and processed data.
- `docs/`: thesis text and planning/literature notes.
- `scripts/`: command-line entry points for interpolation and analysis.
- `src/`: reusable Python modules (audio, interpolation, models, and visualization).
- `experiments/`: generated outputs (audio, plots, diagnostics), usually untracked.

Main Entry Points

The most important scripts used in this thesis are:

- `scripts/audio/crossfade.py`: baseline equal-power crossfade.
- `scripts/encodec/interpolation/interpolate.py`: latent interpolation between two bar-aligned tracks.
- `scripts/encodec/interpolation/interpolate_plot.py`: interpolation and visualization in one run.

- `scripts/encodec/analysis/transition_coherence.py`: coherence analysis against the baseline.
- `scripts/encodec/analysis/corr_analysis.py`: latent channel correlation analysis.
- `scripts/encodec/analysis/latent_frequency.py`: per-channel frequency contribution analysis.
- `scripts/encodec/analysis/latent_pca_dtft.py`: PCA/DTFT latent trajectory diagnostics.
- `scripts/qualitative/analyze_survey_responses.py`: listening-test response summaries.

Reproducibility Note

Generated artifacts such as rendered transitions, spectrograms, PCA plots, coherence plots, and animations are written to `experiments/`. These files are typically large and are therefore not version-controlled, but rather reproduced by re-running the scripts with the same inputs and settings. The exact scripts used for the main evaluation of this thesis is available in the `scripts.txt` file.