FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Video-based Music Generation

Serkan Sulun

Doctoral Program in Electrical and Computer Engineering

Supervisor: Dr. Paula Viana

Supervisor: Dr. Matthew E. P. Davies

May 23, 2025

Video-based Music Generation

Serkan Sulun

Doctoral Program in Electrical and Computer Engineering

Approved in oral examination by the committee:

President: Dr. João Paulo Trigueiros da Silva Cunha
Faculty of Engineering, University of Porto

Member: Dr. Rui Pedro Pinto de Carvalho e Paiva
Faculty of Science and Technology, University of Coimbra

Member: Dr. Tom Collins
Frost School of Music, University of Miami

Member: Dr. Sofia Carmen Faria Maia Cavaco
Faculty of Science and Technology, NOVA University Lisbon

Member: Dr. Maria Teresa Magalhães da Silva Pinto de Andrade
Faculty of Engineering, University of Porto

Member (Supervisor): Dr. Paula Maria Marques de Moura Gomes Viana
ISEP, Polytechnic of Porto

May 23, 2025

Abstract

As the volume of video content on the internet grows rapidly, finding a suitable soundtrack remains a significant challenge. This thesis presents EMSYNC (EMotion and SYNChronization), a fast, free, and automatic solution that generates music tailored to the input video, enabling content creators to enhance their productions without composing or licensing music, streamlining creativity and production. Our model creates music that is emotionally and rhythmically synchronized with the video, offering an adaptive and expressive solution for automatic soundtrack generation.

A core component of EMSYNC is a novel video emotion classifier. To achieve accurate and efficient video classification, we intelligently fuse pretrained models. We additionally address the data-centric challenges in video classification through cinematic trailer genre classification experiments using a large-scale dataset. By leveraging pretrained deep neural networks for feature extraction and keeping them frozen while training only fusion layers, we reduce computational complexity while improving accuracy. We show the generalization abilities of our method by obtaining state-of-the-art results on Ekman-6 and MovieNet, the largest video datasets for emotion and cinematic genre classification, respectively.

Another key contribution is a large-scale, emotion-labeled MIDI dataset for affective music generation. Using annotations from online resources, we build the largest MIDI dataset with valence-arousal labels. We additionally analyze the emotional content of song lyrics within the MIDI files. We then present an emotion-based MIDI generator, the first to condition on continuous emotional values rather than discrete categories, enabling nuanced music generation aligned with complex emotional content.

To enhance temporal synchronization, we introduce a novel temporal boundary conditioning method, called "boundary offset encodings," aligning musical chords with scene changes. Integrated into EMSYNC, this method ensures music naturally follows the video's pacing and rhythm, improving the overall user experience.

We also explore audio synthesis, focusing on audio bandwidth enhancement due to the scarcity of paired MIDI-audio data. We present a proof-of-concept to highlight and address the challenges in audio synthesis, emphasizing generalization. For the first time, we identify the problem of "filter overfitting," where models trained on specific low-pass filters fail to generalize to real-world scenarios. To address this, we propose a data augmentation strategy that outperforms standard regularization methods, marking the first step toward developing robust audio enhancement models for real-world use.

Combining video emotion classification, emotion-based music generation, and temporal boundary conditioning, EMSYNC emerges as a fully automatic video-based music generator. User studies show that it consistently outperforms existing methods in terms of music richness, emotional alignment, temporal synchronization, and overall preference. As a result, EMSYNC sets a new state-of-the-art in video-based music generation, creating music that is both emotionally and rhythmically aligned with the video.

Resumo

A disponibilização de conteúdos vídeo na internet tem crescido de forma exponencial colocando um acervo significativo, e de valor elevado, disponível à população em geral. No entanto, para os produtores destes conteúdos, encontrar uma banda sonora adequada continua a ser um desafio considerável. Esta tese apresenta o EMSYNC (EMotion and SYNChronization), uma solução rápida, gratuita e automática que gera música adaptada ao vídeo de entrada, permitindo aos criadores de conteúdo melhorar as suas produções sem necessidade de compor ou licenciar música, facilitando assim a criatividade e a produção. O nosso modelo cria música emocional e ritmicamente sincronizada com o vídeo, oferecendo uma solução adaptativa e expressiva para geração automática de bandas sonoras. Um dos componentes centrais do EMSYNC é um novo classificador de emoções em vídeo. Para alcançar uma classificação precisa e eficiente, propomos uma fusão inteligente de modelos previamente treinados. De forma a ultrapassar os desafios associados à escassez de dados adequados a esta tarefa, e atendendo aos objetivos similares, tiramos partido das grandes coleções de dados associados a tarefas de classificação de géneros de trailers cinematográficos. Ao explorar redes neuronais profundas pré-treinadas para extração de características e mantendo-as congeladas enquanto treinamos apenas as camadas de fusão, reduzimos a complexidade computacional e melhoramos a precisão. Demonstramos as capacidades de generalização do nosso método ao ultrapassar os atuais resultados estado da arte nos conjuntos de dados Ekman-6 e MovieNet, os maiores conjuntos de dados de vídeo para classificação emocional e de género cinematográfico, respetivamente.

Outra contribuição relevante desta tese é um conjunto de dados MIDI anotado com emoções. Combinando informações recolhidas de recursos online e a análise do conteúdo emocional das letras das músicas, construímos o maior conjunto de dados MIDI com anotações de valência-excitação, disponibilizando assim um recurso poderoso para treinar modelos para geração de música baseados em emoções. Um modelo capaz de gerar MIDI baseado em emoções, o primeiro a ser condicionado por valores emocionais contínuos em vez de categorias discretas, é outra das contribuições fundamentais da tese. Esta abordagem permite a geração de música alinhada com conteúdos emocionais complexos. Para melhorar a sincronização temporal entre os conteúdos de áudio e de vídeo, introduzimos um novo método de condicionamento de fronteiras temporais, denominado "codificação de desfasamento de fronteira", que alinha acordes musicais com mudanças de cena no vídeo. Integrado no EMSYNC, este método garante que a música segue naturalmente o ritmo e o andamento do vídeo, melhorando a experiência do utilizador.

A tese explora ainda a síntese de áudio. Dada a escassez de associação de dados MIDI-áudio multi-instrumento, focamo-nos em aspetos relacionados com a melhoria da largura de banda. Apresentamos uma prova de conceito que permite identificar e responder aos principais desafios associados à síntese áudio, com especial foco na capacidade de generalização. Identificamos o problema de "overfitting de filtros", em que modelos treinados com filtros passa-baixo específicos não generalizam bem para cenários do mundo real. Para ultrapassar este problema, propomos uma estratégia de aumento de dados que supera os métodos de regularização padrão, representando o

primeiro passo para o desenvolvimento de modelos robustos de melhoria de áudio para uso real.

Combinando classificação emocional de vídeo, geração musical baseada em emoções e condicionamento temporal, o EMSYNC afirma-se como um gerador musical automático baseado em vídeo. Estudos com utilizadores demonstram que supera consistentemente os métodos existentes em termos de riqueza musical, alinhamento emocional, sincronização temporal e preferência geral. Assim, o EMSYNC estabelece um novo estado da arte na geração musical baseada em vídeo, criando música que se alinha emocional e ritmicamente com o vídeo.

Acknowledgements

I would like to begin by expressing my sincere gratitude to my supervisors, Paula Viana and Matthew Davies, for their continual support and guidance. I am also forever grateful to my family—Ali, Lale, and Zeynep—for their unconditional love and support. They have provided me with a solid foundation, allowing me to build an independent life without the weight of childhood traumas on top of the usual pains of adulthood.

People often say that relationships during a PhD are difficult, but having an incredible girlfriend has made this period much easier for me. I can never thank my love, companion, and best friend Chloé enough. In this foreign country, she has given me a new home.

A special thanks goes to Bianca, not only for being an amazing friend but also for setting me up with teaching jobs and, most importantly, for tolerating my complaints. I also want to thank my good friends and fellow academics in Porto—Göksu, Nathalie, and Nabila—for their unwavering support, friendship, and valuable advice. I furthermore thank you all for welcoming me into the "Porto Crew."

I am incredibly lucky to have my great friend Doğan, with whom I've maintained a close friendship for the past 20 years, despite living in different countries. I am grateful to my colleagues and office mates—Inês, Luís, Eduardo, Américo, and Tiago—for their friendship and support, helping me navigate my time in Portugal.

Finally, I want to express my thanks to my fellow PhDs from the la Caixa Fellowship cohort—Gizem, Marta, Inês, Arturo, Javi, Chris, Jacob, Paula, Jaime, and Ludovica—for their friendship and organizing some unforgettable holidays.

I am also grateful to have secured two fellowships, namely the la Caixa Doctoral Fellowship (INPhINIT) (LCF/BQ/DI19/11730032) and the FCT (Fundacao para a Ciencia e a Tecnologia) Doctoral Fellowship (2022.09594.BD), that enabled me to pursue this doctoral research. Additional support was provided by funding from the NEXUS-2 Project (12407/BI-M-ED_B2/2025).

Serkan Sulun

Contents

List of Figures

List of Tables

List of Acronyms

AAC	Affective Algorithmic Composition
AEC	Acoustic Echo Cancellation
API	Application Programming Interface
ASR	Automatic Speech Recognition
AST	Audio Spectrogram Transformer
BERT	Bidirectional Encoder Representations from Transformers
BEATs	Bidirectional Encoder representation from Audio Transformers
BN	Batch Normalization
BPM	Beats Per Minute
CLIP	Contrastive Language-Image Pretraining
CMT	Controllable Music Transformer
CNN	Convolutional Neural Network
CTRL	Conditional TRansformer Language
DA	Data Augmentation
DAW	Digital Audio Workstations
DSD	Demixing Secrets Dataset
DNN	Deep Neural Network
DO	DropOut
EMSYNC	EMotion and SYNChronization
EMOPIA	EMOtion PIAno
FAD	Fréchet Audio Distance
FC	Fully-connected
GPT	Generative Pre-Training
GRU	Gated Recurrent Unit
I3D	Two-Stream Inflated 3D ConvNets
ICASSP	International Conference on Acoustics, Speech, and Signal Processing
LLM	Large Language Model
LMD	Lakh MIDI Dataset
LPC	Linear Predictive Coding
LPD	Lakh Pianoroll Dataset
LSTM	Long Short-Term Memory

MAC	Multiply-ACcumulate
mAP	Mean Average Precision
MAESTRO	MIDI and Audio Edited for Synchronous TRacks and Organization
MIDI	Musical Instrument Digital Interface
MLP	Multi Layer Perceptron
MSD	Million Song Dataset
MVED	Music Video Emotion Dataset
NaN	Not A Number
NLL	Negative Log-likelihood
NLP	Natural Language Processing
OCR	Optical Character Recognition
P	Precision
PERR	Pairwise Emotional Relationship Recognition
PESQ	Perceptual Evaluation of Speech Quality
R	Recall
ResNet	Residual Network
RNN	Recurrent Neural Network
SNR	Signal-to-Noise Ratio
SR	Super-Resolution
TRN	Temporal Relation Network
TSN	Temporal Segment Network
VEMOCLAP	Video EMOtion CLassifier using Pretrained features
VGMIDI	Video Game Musical Instrument Digital Interface
VGG	Visual Geometry Group
ViSQOL	Virtual Speech Quality Objective Listener
ViT	Vision Transformer
ViViT	Video Vision Transformer
YOLO	You Only Look Once

Chapter 1

Introduction

The exponential growth of user-generated multimedia content is driven by the increasing availability of affordable, high-quality recording equipment and video editing software (Falkowski-Gilski & Uhl, 2020). A major challenge in content creation is selecting suitable soundtracks to enhance viewer engagement (Buhler, 2018). However, the unauthorized use of commercially published music infringes copyright, restricting monetization on platforms like YouTube. Alternatives such as purchasing music, hiring composers, or searching for royalty-free tracks can be costly, time-consuming, or fail to ensure proper synchronization with the video. This thesis proposes a fast, cost-free, and royalty-free solution: a model that takes any user-provided video as input and automatically composes a soundtrack that matches the emotional content of the video.

Music can be represented in two primary formats: audio and symbolic. Audio, the standard musical format perceived by general listeners, represents sound as a waveform. Symbolic music, on the other hand, is primarily used by musicians and is exemplified by sheet music, which encodes pitches, durations, and includes performance guides such as crescendo or pianissimo. In the digital domain, symbolic music is represented as a discrete sequence, similar to text. The most widely used digital symbolic music format is MIDI (Musical Instrument Digital Interface). A MIDI file consists of a sequence of events, each encoding information about a note, such as its pitch, instrument (channel), and intensity (velocity/loudness). While a musical composition can be stored as a MIDI file, similar to sheet music, MIDI files must be synthesized into audio to be heard. Software like Fluidsynth[1] assigns sounds to MIDI events using standard libraries, converting them into an audio waveform.

Existing approaches that generate music as audio waveforms lack editability (Copet et al., 2023; Evans et al., 2024; Lam et al., 2023; H. Liu et al., 2024). These methods compress all stages of music production—composition, performance, editing, mixing, and mastering—into a single process, restricting creative control for professionals. In contrast, generating music in a symbolic format such as MIDI provides greater flexibility. Since MIDI explicitly encodes musical elements like notes, timing, and dynamics, professionals can refine compositions, synthesize audio using digital audio workstations (DAWs), or record and adjust performances with ease.

[1] https://fluidsynth.com

Our work focuses on generating music in MIDI format from arbitrary videos using deep neural networks (DNNs). Since MIDI is our exclusive format, we use "MIDI" and "music" interchangeably. A key challenge in training video-to-MIDI DNNs is the lack of large-scale paired video-MIDI datasets. Existing datasets are either domain-specific, such as pianist hand videos (Koepke et al., 2020), or contain only a limited number of samples (around 1k) (Di et al., 2021; Zhuo et al., 2023). While large-scale MIDI datasets like the Lakh MIDI Dataset exist, they are not paired with videos, making end-to-end learning infeasible (Raffel, 2016). Our approach addresses this by identifying intermediate representations that link video and music. This leads to a two-stage framework: first, analysis models extract representations from the input video; then, these representations condition the music generation model.

To realize an automatic soundtrack generator that appeals to human listeners, we must emulate the composition process of human soundtrack composers. Therefore, the intermediate representations should align with how composers create soundtracks. The question of how to compose music that fits a video is longstanding (Buhler, 2018). However, it is an artistic rather than an engineering problem—there is no objective or definitive answer. Nevertheless, certain heuristics and general knowledge suggest that an effective soundtrack should: (1) align with the video's underlying emotions to enhance them (Deutsch, 2007; Schifrin, 2011), and (2) match its temporal dynamics, ensuring synchronization to improve viewer engagement (Cohen, 1990; Prendergast, 1992). Based on this perspective, our method follows a two-branch approach, generating music using both emotional cues and temporal dynamics. Figure 1.1 illustrates our two-stage, two-branch methodology.

The user-provided input video is first analyzed, followed by symbolic music generation based on the analysis output, and finally, the audio that corresponds to the symbolic music is generated. However, in both our work and the structure of this thesis, we adopt a more strategic ordering of these three processes.

In both our research and the narrative of this thesis, we begin with music generation, followed by video analysis. This order ensures that the generated music remains perceptually coherent and musically meaningful at each stage. Designing the video analysis module first would be risky: if the music generator cannot effectively use the extracted representations, we would need to revisit and redesign the video analysis module. Furthermore, once a conditional music generator is in place, even if video feature extraction proves unsuccessful, we could fall back on a contingency plan by getting these features directly from the user.

After designing the music generation and video analysis modules, we focus on integrating them. For MIDI-to-audio generation, we use a standard approach with the dedicated Fluidsynth software and standard sound libraries. While experimental methods based on deep neural networks exist, they are limited to piano tracks and do not generalize to multi-instrument music (Hawthorne et al., 2019; Sambaragi et al., 2024).

Nevertheless, we explore the use of DNNs for audio generation in a different context towards the end of this thesis. This exploration is separate from and not integrated into our main model. We focus specifically on the challenges associated with audio generation. As discussed later in

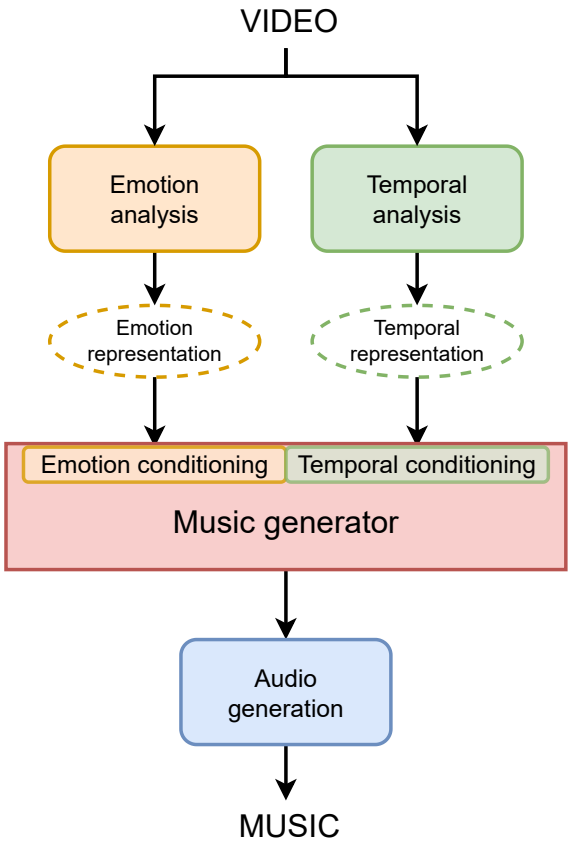


Figure 1.1: Our video-based music generation methodology.

Section 1.4, the lack of datasets containing paired audio and multi-instrumental MIDI forces us to rely on audio-only datasets, particularly for generating high-quality audio from low-quality sources. In this context, we address the problem of audio bandwidth extension, where we create input-target pairs by synthetically band-limiting high-quality audio.

In summary, here is the high-level order of topics we explore throughout this thesis:

1. Symbolic music generation
2. Video analysis
3. Combining the video analysis and music generation modules (video-based music generation)
4. Audio generation and bandwidth extension

We begin our discussion with symbolic music generation.

1.1 Symbolic music generation

Symbolic music, represented as a sequence of notes, is widely used in machine learning models due to its compact and structured format. Large-scale raw MIDI datasets enable unsupervised

training of DNNs for automatic symbolic music generation (Raffel, 2016). The state-of-the-art architecture for sequence generation is the transformer model, which serves as the backbone of all large language models (LLMs) (Vaswani et al., 2017). Similar to language modeling, transformers learn to predict the next token, i.e., the next note, and generate output autoregressively, one token at a time during inference. However, human composers do not simply construct music note by note in a sequential manner; their creative process involves high-level concepts such as motifs, themes, emotions, and temporal structures like rhythm (Meyer, 2008). Inspired by this, we present a *conditional music generator* with two distinct mechanisms to independently incorporate temporal and emotional conditioning.

1.1.1 Temporally-conditioned music generation

We first address the challenge of generating music based on temporal dynamics. However, it is first necessary to define what "temporal dynamics" entail. Previous works on video-based music generation have used motion speed and motion saliency as temporal features (Di et al., 2021). This approach produces a dense representation of temporal dynamics, resulting in continuous conditioning on musical note density. However, our preliminary listening tests of Di et al. (2021) revealed two key limitations. First, the continuously varying note density disrupts rhythmic consistency. Second, because the mapping is dense, synchronization between music and video is difficult to perceive unless there are strong contrasts in motion speed and note density across consecutive sections.

We pose the following research question:

> *Which type of temporal feature is the most useful for synchronizing video and music while maintaining rhythmic consistency?*

Our hypotheses are as follows:

> *Dense temporal representations, such as speed, can result in continuous modifications and disruptions of rhythm.*
>
> *Matching sparse temporal features, such as temporal boundaries, would help preserve a steady rhythm while enhancing the perceived synchronization between video and music.*

Temporal conditioning in MIDI-generating models presents unique challenges. While deep transformer models can process time-based data such as videos, their sequence dimension correlates linearly with time due to a fixed frame rate (Arnab et al., 2021; Z. Liu et al., 2022). In contrast, MIDI is typically represented using an event-based encoding, where sequence and time dimensions are not linearly correlated (Oore et al., 2020).

In event-based MIDI encoding, two primary token types exist: "note" and "time shift." Note tokens represent pitch, while time shift tokens advance the time axis, capturing both note durations and silences. Each time shift token specifies a time increment, creating a one-dimensional

sequence where position in the sequence does not directly correspond to position in time. The key advantage of this encoding is the absence of a fixed time grid, allowing for subtle, expressive timing variations that reflect human musicianship.

We further ask:

> *How can we design methods of temporal synchronization that are independent of token positions, while preserving event-based MIDI encoding?*

We hypothesize:

> *Auxiliary inputs in the form of temporal conditions can facilitate synchronization.*

A particular challenge arises when the model is required to synchronize at a specific timestamp, as forced synchronization may disrupt the consistent rhythm of the generated music. To address this, we further investigate learned conditioning methods with minimal intervention, allowing the model to select the exact synchronization point that aligns harmoniously with the music it generates. Additionally, we hypothesize that providing the model with information about future boundaries can guide it in generating musical notes that precede the boundary in a coherent manner.

1.1.2 Emotion-based music generation

Music has long been a powerful medium for emotional expression and communication (Hunter & Schellenberg, 2010). Its ability to evoke emotions has been studied across various disciplines, including psychology (Krumhansl, 2002), musicology (Juslin & Sloboda, 2001), and neuroscience (Koelsch, 2014). With the rise of deep learning, there is growing interest in machine learning algorithms capable of automatically analyzing and generating music to elicit specific emotional responses in listeners (Briot et al., 2020).

We address the problem of generating music from emotions, aiming to achieve the highest possible performance. Our approach follows the deep learning trend, where larger models trained with more data perform better (Kaplan et al., 2020). However, current MIDI datasets that contain emotion labels are very small, containing fewer than 400 samples (Ferreira & Whitehead, 2019; Hung et al., 2021; K. Zhao et al., 2019). This limitation poses a significant challenge to training large and powerful DNNs for emotion-based MIDI generation.

To address the issue of data scarcity, we explore the following research questions:

> *How can we acquire large-scale datasets consisting of MIDI-emotion label pairs?*
>
> *Is it possible to label existing MIDI datasets with emotion labels without human input?*
>
> *Do other modalities, such as audio or text (e.g., song lyrics), contain information related to emotions?*
>
> *Can we leverage existing music databases and platforms to identify emotion labels and assign them to corresponding MIDI samples?*

We hypothesize that:

> *Emotions that are automatically extracted from other related modalities, particularly from the audio version and built-in song lyrics will align with the corresponding MIDI samples.*

Due to the inherent differences between these modalities, i.e., audio vs. MIDI and text vs. MIDI, these labels will serve as "weak" labels. However, the large volume of available samples is expected to compensate for the weakness of the match between MIDI and emotion labels. Since most musical content available on the internet is in audio format, we aim to explore music platforms to identify metadata related to emotions. We additionally exploit language models to analyze the emotions of the lyrics within the MIDI files.

We now continue with exploring video analysis.

1.2 Video analysis

In video analysis, representations are typically categorized as high-level or low-level based on their abstraction and complexity. Low-level video features are extracted directly from video data without interpreting its content. Examples include pixel-based features such as RGB values and edge information (Canny, 1986), motion features such as optical flow (Horn & Schunck, 1981), texture features like local binary patterns (Ojala et al., 1996), and transition features such as dissolves and scene cuts (Bieda et al., 2021).

High-level video features provide a more abstract interpretation of video content, often captured using machine learning methods. Examples include actions (Jhuang et al., 2013), objects (Redmon et al., 2016), emotions (Soleymani et al., 2012), and genres (Wehrmann & Barros, 2017).

In this work, we utilize both low-level scene cuts and high-level semantic information. Since scene cut detection is a well-established problem, we directly apply existing methods using the FFMpeg software[2]. Our research primarily focuses on semantic video classification, particularly emotion classification, as this is crucial for our emotion-based music generator. Since we aim to develop a music generator that works with any type of video, we specifically focus on the emotion classification of *arbitrary* videos, rather than domain-specific videos, such as those containing dialogues (Gao et al., 2021) or human faces (Lee et al., 2019).

In semantic video classification, as in most video classification tasks, DNNs represent the state of the art. These models primarily use raw data, such as pixel values and audio waveforms, as inputs. Unlike older methods that rely on hand-crafted features, DNNs are designed to implicitly extract high-level features within their layers. However, these models often contain millions of parameters, requiring substantial computational resources, including time, memory, and large-scale training data.

Video classification presents additional challenges due to the sheer size of uncompressed video frame sequences, which further increases computational demands. To address this, prior works

[2]https://ffmpeg.org

have processed only a fixed, limited number of frames or clips (Wehrmann & Barros, 2017; S. Zhao et al., 2020). However, this approach inevitably leads to information loss, potentially compromising classification accuracy. The computational complexity of processing raw videos increases further with the adaptation of larger models—a trend that continues to expand over the years (Bernstein et al., 2021; M. Tan & Le, 2019). When the training data is limited, using larger input dimensionality and larger models also increases the chance of overfitting (Defernez & Kemsley, 1999).

We ask the following research question:

> *How can we design efficient video analysis models with high representation power, potentially capable of handling an arbitrary number of input frames, while ensuring generalization?*

Our hypotheses are:

> *Due to the large size of raw video inputs, training models end-to-end on such data is prohibitively expensive and likely inefficient.*
>
> *Models pretrained on various multimedia tasks can provide compact and meaningful inputs for video classification, leading to improved performance and efficiency.*

Using pretrained models brings multiple advantages. Firstly, since the pretrained model is used in inference mode, its weights remain frozen, therefore reducing the time and memory complexity by requiring only a forward pass, without an additional backward pass. Secondly, it considerably reduces the size of the input fed to the subsequent model that is being trained. As an example, a very small video frame contains $224 \times 224 \times 3 = 163{,}968$ values, while the state-of-the-art image analysis model CLIP (Contrastive Language-Image Pre-Training) represents an image using only 512 values (Radford et al., 2021). This size reduction becomes more important as the duration of the video increases. Thirdly, assuming that the pretrained features of multimedia analysis models present valuable and relevant information for the task of video classification, their use removes the need to train new models to extract similar features implicitly from raw data. This reduces the size of the subsequent model that needs to be trained. Finally, since it reduces the input size and the number of weights in training, it also reduces the chance of overfitting (Defernez & Kemsley, 1999).

There are several methods to utilize pretrained networks, which have significant overlaps. These methods are commonly referred to as transfer learning, fine-tuning, or using pretrained features (Niu et al., 2020; C. Tan et al., 2018). These terminologies can be distinguished as follows: Transfer learning involves applying a model trained on one task to a different task (Weiss et al., 2016). Fine-tuning entails further training the transferred model for the new task (Church et al., 2021). In our work, we use pretrained features, which involves performing inference on the data using a pretrained model, and then feeding the resulting outputs or activations (features) into a new model that is trained (Reiss et al., 2021). These methods are frequently employed in video

processing tasks such as human activity recognition (Ray & Kolekar, 2024), video summarization (Khan et al., 2024), and video recommendation (A. Almeida et al., 2022).

We continue with the following research questions:

> *How does training a task-specific DNN end-to-end compare with utilizing frozen pretrained models?*
>
> *How can we efficiently use open-source pretrained networks and combine them for video classification?*

Our hypothesis is as follows:

> *Powerful multimodal pretrained models can be used in inference mode, and their outputs can be fused with lightweight neural networks trained separately—enhancing both performance and efficiency.*

We present architectures trained end-to-end and models that fuse pretrained networks, and then compare their performance. We employ pretrained models for image analysis, optical character recognition, automatic speech recognition, audio event classification, music classification, and facial emotion classification to incorporate information about visual scenery, objects, text, speech, music, audio, and human faces. These pretrained models are used in inference mode, meaning they are not fine-tuned, which results in significantly lower time and memory requirements. We then use trained neural networks to fuse the pretrained features and obtain the final classification results. Since pretrained models extract meaningful features from raw data, a shallow fusion network with just one or two layers is sufficient as the classifier, greatly reducing computational complexity, hardware requirements, and training time.

We also investigate various semantic video classification datasets in terms of quality, quantity, and their suitability for our overall objective. We explore the types of video datasets that offer the most variety to enable our models to generalize to arbitrary videos. We hypothesize that user-generated videos and cinematic videos are the two best candidates. User-generated videos, collected from platforms such as YouTube[3] and Flickr[4], have minimal restrictions on the type of videos uploaded, aside from legal considerations. Similarly, cinema, as an art form, seeks to push the boundaries of creativity and, therefore, contains a wide range of visual and narrative variety.

Our preliminary investigations reveal a lack of high-quality, large-scale video datasets with emotion labels. Ekman-6, one of the largest video emotion classification benchmarks, only contains 1,637 videos and leads to overfitting (B. Xu et al., 2018). Using a small, fixed number of input frames addresses overfitting but at the cost of information loss (Wehrmann & Barros, 2017; S. Zhao et al., 2020). In this work, we aim to tackle these issues of data scarcity and information loss. We pose the following research questions:

> *What other semantic video classification tasks can provide a large amount of training data to further mitigate overfitting?*

[3] https://youtube.com
[4] https://flickr.com

Given sufficient data, what is the relationship between the number of input frames and classification performance?

We further challenge the notion of using a small and fixed number of frames. We argue that:

Cinematic video datasets provide a vast amount of labeled video samples, thanks to the availability of online catalogs such as IMDb[5] *and video hosting platforms like YouTube (Harper & Konstan, 2016; Q. Huang et al., 2020).*

By using compact inputs and a large amount of data, increasing the number of input frames will consistently improve classification performance.

To this end, we explore cinematic datasets that offer a substantial number of samples. By leveraging their size, we can address overfitting, train larger and more powerful models, and utilize more input frames to fully maximize the information provided to the model.

A cinematic counterpart to video emotion classification is movie trailer genre classification. Since both emotion and genre classification involve discrete categories, we can employ classifier models that predict probability distributions over a set of predefined labels. This similarity enables us to tackle both tasks using a shared model architecture.

We continue our discussion with the integration of the symbolic music generation and video analysis modules to realize a video-based music generator.

1.3 Video-based music generation

We finally integrate the previously built components—video emotion classifier, FFMpeg scene cut extractor, emotion-conditioning mechanism, and temporal boundary conditioning mechanism—to construct the video-based music generator.

A key challenge arises from the mismatch in representation between the video emotion classifier and the emotion-conditioning mechanism of the music generator. While our video emotion classifier outputs probabilities for discrete emotions, our music generator represents emotions using continuous valence-arousal values (Russell, 1980). We ask:

How can we bridge categorical (discrete) and dimensional (continuous) emotion representations?

We hypothesize that:

Statistical data from psychological user studies may support the development of a mapping method between these two representations.

To bridge this gap, we introduce a simple mapping method based on psychological studies (Russell & Mehrabian, 1977). These studies present participants with scenarios associated

[5] https://imdb.com

with discrete emotions and ask them to rate the felt emotion in terms of valence (pleasantness vs. unpleasantness) and arousal (intensity). Researchers then report the mean and standard deviation of valence and arousal for each discrete emotion category. We use these statistical values to construct a mixture of Gaussian distributions, where each component corresponds to a discrete emotion. The final valence-arousal value is then sampled from this mixture, weighted according to the probability distribution output by the video emotion classifier. Finally, we evaluate our complete model against existing video-based music generators. Extensive user studies demonstrate that our method outperforms prior approaches across all evaluated metrics.

While the audio samples in our user study are generated using off-the-shelf Fluidsynth software, we further explore the task of audio generation—focusing particularly on bandwidth extension and its associated challenges.

1.4 Audio generation and bandwidth extension

Our overall video-based music generator outputs symbolic music that is compact, editable, and interpretable. To generate audio that can be heard by humans, the users have several alternatives, ranging from quick but low-quality methods to high-quality but time-consuming approaches. First, users can utilize computer software such as Fluidsynth and soundfont libraries to quickly synthesize MIDI into audio with minimal intervention. These soundfont libraries provide fixed sounds for specific pitches of particular instruments. This is the approach we use to generate our final samples, as our work is focused on music composition rather than sound design. However, as expected, the resulting audio is mechanical, unrealistic, and generally of lower quality compared to music produced by human musicians.

Another alternative is to use Digital Audio Workstations (DAWs) to generate audio from MIDI. In this approach, the end user can edit all aspects of the MIDI, including notes, tempo, instruments, and velocities, as well as the audio itself, using proprietary soundfonts, custom effects, and various mixing and mastering techniques. While this approach produces higher-quality audio, it is manual, requiring significant time, skills, and labor. Finally, human musicians can perform and record the compositions generated by our music generator. This method yields the highest-quality audio but also demands the most time, skills, and labor.

An alternative research-oriented solution is to use DNNs to synthesize audio from MIDI. However, this is an ill-defined problem, as the same music composition, i.e., a MIDI file, can have an infinite number of human performances. There are models that attempt to tackle this task, but they rely on paired datasets of MIDI and audio, which are typically piano-based (Hawthorne et al., 2019; Sambaragi et al., 2024). Since our work focuses on multi-instrumental music generation, we face the challenge that there are no multi-instrument datasets with pairs of audio performed by humans and MIDI. One could train DNNs on synthetic datasets containing MIDI and the corresponding synthesized audio, but this would limit the DNN's performance to the quality of the

synthesis method, rendering it somewhat redundant. Additionally, this could be viewed as a generalization problem, where the model only generates synthetic-sounding audio and fails to generalize to real-world samples created by musicians.

To investigate this problem further, we ask the following research questions:

> *What other audio generation tasks exhibit overfitting caused by synthetic data?*
>
> *How can we quantify and alleviate overfitting to synthetic data in audio generation tasks?*

Our hypotheses are as follows:

> *Audio enhancement tasks also suffer from overfitting, due to their use of synthetically degraded audio as inputs.*
>
> *To systematically analyze overfitting, and to mitigate it, we can utilize a diverse set of degradation functions during the training and testing of DNNs.*

Audio enhancement tasks involve using low-quality audio as input and its high-quality counterpart as output. Audio enhancement models are often trained with synthetic data, where high-quality audio is degraded using digital filters to create the low-quality counterparts (Kuleshov et al., 2017). This task only requires a high-quality audio dataset, without the need for a corresponding sample from a different modality, such as MIDI. As audio enhancement tasks only use audio-only samples, the data is abundant. We aim to quantify generalization by applying multiple digital degradation filters and measuring whether the trained models generalize across different filters. Specifically, we focus on the task of audio bandwidth extension, which allows for both qualitative and quantitative analysis by observing and evaluating the reconstruction quality of the missing frequency bands. Furthermore, in audio bandwidth extension, we can use various low-pass filters by varying their types and orders.

As this problem stems from the use of synthetic data, we hypothesize the solution is to use data augmentation strategies, and not model-based methods. Our results show that using a variety of filters while creating the input samples alleviate overfitting. Our approach marks the first step towards achieving real-world audio bandwidth extension models.

We now summarize our contributions.

1.5 Contributions

Our first key contribution is the construction of a large-scale MIDI dataset with emotion labels. This dataset enables training models where emotions serve as conditional inputs and MIDI sequences act as ground-truth targets. Motivated by the well-established principle that larger models trained on larger datasets yield better results (Sevilla et al., 2022), we avoid using existing emotion-labeled MIDI datasets, as they contain few samples (Ferreira & Whitehead, 2019; Hung et al., 2021; K. Zhao et al., 2019). Instead, we assign emotion labels to the Lakh MIDI dataset, which

comprises 176,581 samples (Raffel, 2016). By integrating multiple auxiliary datasets, we annotate 34,791 MIDI samples with valence and arousal values, creating the largest emotion-labeled MIDI dataset to date. Using this dataset, we successfully train large transformer models (Vaswani et al., 2017) to generate multi-instrument symbolic music conditioned on emotion. To the best of our knowledge, this is the first music generation model capable of conditioning on both valence and arousal simultaneously, allowing control over any emotion within the widely used circumplex model of affect (Russell, 1980). We shared our findings in the following paper and open-source code.

> ***S. Sulun**, M. E. P. Davies, and P. Viana, "Symbolic Music Generation Conditioned on Continuous-Valued Emotions," IEEE Access, vol. 10, pp. 44617–44626, 2022.*
>
> *Emotion-based MIDI generator.* https://github.com/serkansulun/midi-emotion

We additionally provide emotion labels for MIDI files by classifying the sentiments of lyrics contained within the MIDI files. To achieve this, we first train emotion classification models on GoEmotions, one of the largest text datasets with 28 fine-grained emotion labels (Demszky et al., 2020). Using these models, we infer emotion labels from the lyrics associated with the MIDI files. Although this approach did not succeed in improving emotion-based MIDI generation, our model achieved state-of-the-art performance on the GoEmotions test set while being only half the size of the baseline model. We presented our findings in the following paper and open-source code and dataset.

> ***S. Sulun**, P. Oliveira, and P. Viana, "Emotion4MIDI: A Lyrics-Based Emotion-Labeled Symbolic Music Dataset," in Progress in Artificial Intelligence, Cham: Springer Nature Switzerland, 2023, pp. 77–89.*
>
> *Emotion classification of MIDI lyrics. Includes pretrained emotion text classifier, and dataset with the labels for the MIDI files in Lakh and Reddit MIDI datasets.* https://github.com/serkansulun/lyricsemotions

We also present a novel mechanism for the temporal conditioning of our music generator. Unlike state-of-the-art video-based music generators that rely on fixed, coarse time grids (Di et al., 2021; Kang et al., 2024), our method preserves event-based encoding while enabling fine-grained temporal control. We achieve this by encoding and feeding *boundary offsets* into the model. These offsets inform the model of the remaining time until the next chord, allowing it to "anticipate" upcoming chords and generate preceding notes accordingly. As a result, the generated chords and surrounding notes remain rhythmically and harmonically coherent. We introduced this method in our final paper that presents our overall model, along with an online demo.

> ***S. Sulun**, P. Viana, and M. E. P. Davies, "Video Soundtrack Generation by Aligning Emotions and Temporal Boundaries," arXiv: arXiv:2502.10154.*
>
> *Demo.* https://colab.research.google.com/drive/1uPLfNZjlvUyMNuH3ky-JmGUaNbkbpmDX

We then present our approach for semantic video classification. Our method maintains relatively low computational complexity by leveraging compact pretrained features instead of raw frames. In video emotion classification, we fuse the pretrained features intelligently using cross-attention layers (Bahdanau et al., 2015). Our method outperforms state-of-the-art models in one of the largest video emotion classification datasets, Ekman-6 (B. Xu et al., 2018), by a significant margin. As a secondary contribution, we identify labeling errors within the Ekman-6 dataset. These errors stem from issues in the video search process, such as mislabeling videos of celebrities named "Joy" as expressing the emotion of joy. To address this, we manually review all videos in the Ekman-6 dataset and compile a blacklist of mislabeled samples to assist future researchers. We presented our findings in the following paper and online demo:

> ***S. Sulun**, P. Viana, and M. E. P. Davies, "VEMOCLAP: A video emotion classification web application," International Symposium on Multimedia (ISM), IEEE Computer Society, pp. 137–140, 2024.*
>
> *Video emotion classifier.* https://github.com/serkansulun/video-emotion
>
> *Demo.* https://colab.research.google.com/drive/1S-Nsm968-KTErbuU0qHOp7mEEzh6xOVx

While the Ekman-6 dataset contains 1,637 videos averaging 10 seconds in length (B. Xu et al., 2018), the cinematic dataset MovieNet consists of 33,000 movie trailers, each averaging 2 minutes (Q. Huang et al., 2020). For trailer genre classification using MovieNet, we train more robust architectures by incorporating a greater number of input frames without risking overfitting. To fuse pretrained features extracted from both video frames and audio segments, we employ the transformer architecture—recognized as the state-of-the-art for sequence processing and the backbone of modern large language models (Vaswani et al., 2017). This architecture is well-suited for handling long input sequences and capturing long-term dependencies. Leveraging this capability, our genre classification model processes the full video and audio content: it first identifies scene cuts and then uses *all* the resulting frames, alongside the complete audio. Consequently, our approach outperforms existing state-of-the-art models in genre classification on MovieNet, the largest cinematic dataset available. Additionally, we demonstrate empirically that classification performance improves consistently as more input frames are incorporated. We detail our findings in the following paper and provide the corresponding open-source code.

> ***S. Sulun**, P. Viana, and M. E. P. Davies, "Movie trailer genre classification using multimodal pretrained features," Expert Systems with Applications, vol. 258, p. 125209, 2024.*
>
> *Trailer genre classifier.* https://github.com/serkansulun/trailer-genre-classification

We finally raise the issue of filter generalization for DNNs neural networks applied to musical audio bandwidth extension. Contrary to many problems for which deep learning is used, we do not find any evidence of overfitting to audio samples themselves (i.e. the training data), but rather,

we observe a clear trend for state of the art DNNs to overfit to filter shapes. When these DNNs are presented with audio excerpts that have been preprocessed with low pass filters not included in the training, then no meaningful extension of the bandwidth can be obtained. Furthermore, the use of widely adopted regularization layers such as batch normalization and dropout fall short in alleviating this problem. To address the filter overfitting issue we propose a novel data augmentation approach, which uses multiple filters at the time of training. Our results demonstrate that without data augmentation, filter overfitting increases as training progresses, whereas including data augmentation is a promising step towards achieving filter generalization. Our findings are presented in the following paper and open-source code.

> ***S. Sulun*** *and M. E. P. Davies, "On filter generalization for music bandwidth extension using deep neural networks," IEEE Journal of Selected Topics in Signal Processing, vol. 15, no. 1, pp. 132–142, 2020.*
>
> *Audio bandwidth enhancer.* https://github.com/serkansulun/deep-music-enhancer

The remainder of this thesis is structured as follows. Chapter 2 introduces the background and related work on emotion representations, transformers, sequence processing (text and MIDI), video analysis, video-based music generation, and audio bandwidth extension. Chapter 3 details our approach to labeling a large MIDI dataset with emotions. Chapter 4 discusses our conditional music generation methods, focusing on emotion and temporal boundary conditioning. Chapter 5 presents our video analysis models, specifically for emotion classification, genre classification, and scene cut extraction. Chapter 6 integrates these components into a complete model for video-based music generation. Chapter 7 presents our exploration on the task of audio bandwidth extension. Finally, Chapter 8 summarizes our conclusions and outlines future directions.

Chapter 2

Background and related work

In this chapter, we begin by introducing the key concepts related to our goal of video-based music generation. Our work involves multiple types of data—video, image, audio, text, symbolic music, and emotion. While video, image, audio, and text have well-known and standardized digital formats, symbolic music and emotion do not have a single common way of being represented. We therefore start by presenting the symbolic music and emotion representations, and the different ways that they are encoded.

Next, we introduce the transformer architecture, which is the state-of-the-art approach for processing sequences (Vaswani et al., 2017). Since symbolic music is a sequence of notes, video is a sequence of frames, and text is a sequence of words or subwords, we rely heavily on transformers across the various subtasks in our work. Later in the thesis, we present significant modifications to the standard transformer to implement novel conditioning mechanisms as well as for multimodal processing. For this reason, it is important to first provide a theoretical background on how transformers work.

After explaining aforementioned key concepts, we present a literature review on specific applications. We first describe the generic task of sequence processing and then its particular subtasks such as text emotion classification, conditional generic sequence generation, and conditional symbolic music generation.

We then cover video analysis in a dedicated section, separate from generic sequence processing. Although video can be described as a sequence of frames, its structured pixel layout—spanning width, height, and time—distinguishes it from tokenized sequences like MIDI and text. In this section, we also discuss video emotion classification and trailer genre classification.

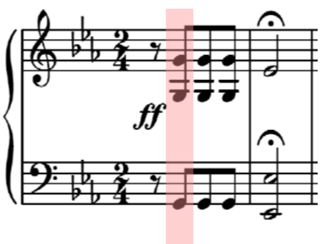
Next, we review the literature on video-based symbolic music generation, which represents the current state-of-the-art in our overall task. We finally present the works on audio bandwidth extension, which is a subtask of audio generation.

2.1 Symbolic music

Symbolic music formats, such as MIDI (Musical Instrument Digital Interface), are used to represent musical performances and compositions in the digital domain. These files contain only the musical information, such as notes, tempo, and dynamics, without the actual sound, making them analogous to a "digital music sheet." Compared to audio formats, symbolic music files are much smaller in size and dimensionality, which makes them more manageable and suitable for modeling with deep neural networks (Briot et al., 2020). The two prevailing music formats are MIDI (Musical Instrument Digital Interface) and pianoroll, which can seamlessly be converted into each other.

Unlike audio formats (e.g., WAV, MP3), MIDI does not store actual sound. Instead, it encodes musical instructions, such as pitch, velocity, duration, and instrument choice, which can be interpreted by MIDI-compatible devices (e.g., synthesizers, virtual instruments). MIDI files consist of *events*, where each note is encoded with a delta-time value (denoting the time shift), pitch (denoting the note), channel (denoting the instrument), and velocity (denoting the loudness). This results in a 1-dimensional sequence representation.

In contrast, the pianoroll format organizes the notes in a 2-dimensional representation: the horizontal axis represents time, the vertical axis represents pitch, and the values correspond to velocity, resulting in a 2-dimensional matrix. For multi-instrument music, multiple 2-dimensional matrices are used—one per instrument. Figure 2.1 displays the first twelve notes of Beethoven's Fifth Symphony, represented in sheet music, MIDI, and pianoroll formats. In this figure, the sheet music and the MIDI represents the left and right hand notes of the piano using different staves and channels, respectively. The pianoroll collects them into a single matrix since they belong to the same instrument.



Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	63	100
0	NOTE ON	2	51	100
0	NOTE ON	2	39	100
240	NOTE OFF	1	63	0
0	NOTE OFF	2	51	0
0	NOTE OFF	2	39	0

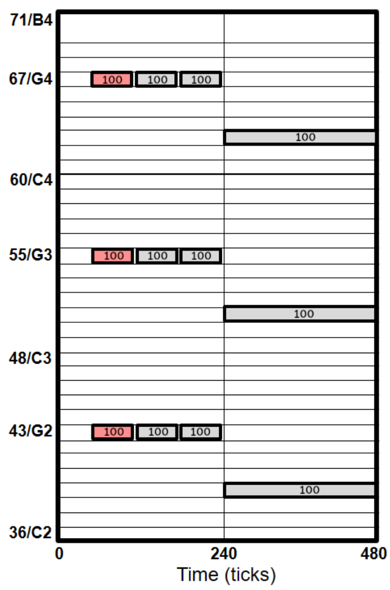


Figure 2.1: Sheet music (left), MIDI (middle), and pianoroll (right) representations. Taken from Müller (2015).

The *event-based* symbolic music representation is an adaptation of MIDI for neural network use (Oore et al., 2020). It splits some of the MIDI properties into separate events before tokenization, creating a dictionary of manageable size. In particular, it separates the delta-time information to create `TIME-SHIFT` events and the velocity information to create `SET-VELOCITY` events. `TIME-SHIFT` tokens are used to move along the time axis, representing both note durations and the silences between them. Each token specifies a time increment in milliseconds. For example, an 800-millisecond shift is encoded as `TIME-SHIFT<800ms>`. Both velocities and time shifts can be quantized to reduce the vocabulary size. Typically, time shifts are quantized at 8 milliseconds, with a maximum time shift of 1000 milliseconds (Oore et al., 2020). Timeshifts longer than 1000 milliseconds can be represented by multiple consecutive time shift tokens.

Figure 2.2 displays the conversion of symbolic music into an event-based representation. Pianoroll is used in this figure as it is a more visual representation compared to MIDI, as seen in 2.1. However, the same conversion process applies to MIDI as well. First, a C4 note with a duration of 640 milliseconds and a velocity of 31 is played, followed by a 24-millisecond silence. After that, an F3 note is played with a velocity of 25. When processing the first note, assuming the previous velocity was different, we set the velocity to 31 using a `SET-VELOCITY<31>` token. Then, we mark the beginning of the first note and its velocity using a `NOTE-ON<C4>` token. Next, using the `TIME-SHIFT<640ms>`, we move forward in time by 640 milliseconds, until we arrive at the next note boundary (i.e., a `NOTE-ON` or a `NOTE-OFF` token). Since the velocity of the next note differs from the currently set velocity of 31, we set the new velocity to 25 using a `SET-VELOCITY<25>` token. Finally, we mark the beginning and the pitch of the next note using a `NOTE-ON<F3>` token.

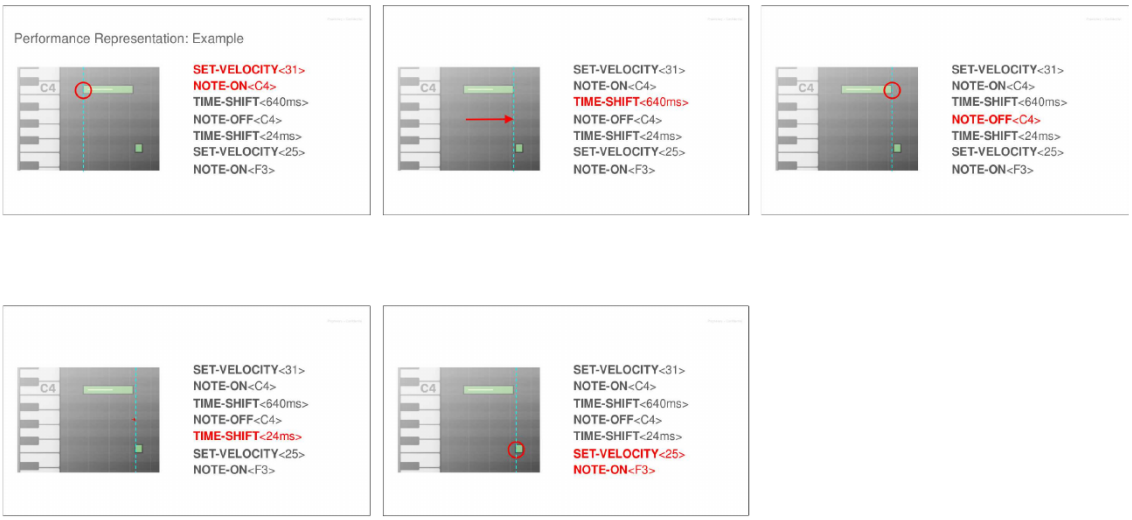


Figure 2.2: Conversion of pianoroll into event-based representation. Taken from Oore et al. (2020). License: CC 4.0 BY

Event-based encoding was originally developed for single-instrument music, particularly piano music (Oore et al., 2020). However, it is straightforward to encode instrument information by injecting it into the `NOTE-ON` and `NOTE-OFF` tokens (Donahue et al., 2019). This modification

allows for the representation of multi-instrument music, though it increases the vocabulary size. As a result, we obtain tokens such as `NOTE-ON-PIANO<C4>` and `NOTE-OFF-GUITAR<F3>`.

We now turn to the next modality relevant to our work: emotions.

2.2 Emotion representations

Emotion is a central concept supporting multiple subtasks in this thesis. It forms one of the two branches of our video-based music generator, serving as the bridge that links video content to musical output. We employ emotions as targets and outputs while classifying videos and as input conditions while generating music. However, unlike modalities such as video, which is always represented as an array of pixels, emotions can be modeled in multiple ways, making their computational modeling less standardized.

Emotion representation is a fundamental concept in affective computing and psychology, forming the basis for analyzing and modeling human emotions (Calvo & Mac Kim, 2013; Matsuda et al., 2013; Morgan, 2019). There are two primary approaches to representing emotions: categorical and dimensional models.

Ekman (1971) presented a categorical approach to classify facial expressions and physiological responses using discrete categories, such as happiness, sadness, anger, fear, surprise, and disgust. The dimensional approach represents emotions as points within a continuous space, typically defined by valence (ranging from unpleasant to pleasant) and arousal (ranging from calm to excited) (Russell, 1980). Both approaches offer distinct strengths and limitations. Categorical models align more naturally with how humans perceive and label emotions, offering simplicity and clarity. Yet, their reliance on a fixed set of discrete categories results in a coarse representation that may overlook the fluid and overlapping nature of emotional experiences. In contrast, dimensional models provide a nuanced and flexible representation of emotional states, capturing subtle variations through continuous numerical coordinates. However, this complexity can make them less intuitive and harder for the general public to interpret.

2.2.1 Categorical emotion representation

Ekman (1971) conducted pioneering research on facial expressions and emotions, identifying six universal facial emotions that are present across cultures: happiness, sadness, anger, fear, surprise, and disgust. These categories continue to serve as labels for modern emotion-related datasets (B. Xu et al., 2018). Expanding on the role of emotions in early development and psychological functioning, Izard (2013) proposed a broader set of ten basic emotions: joy, interest, sadness, anger, disgust, contempt, fear, shame, guilt, and surprise.

Plutchik (1988) introduced an alternative framework by organizing eight primary emotions—joy, trust, fear, surprise, sadness, disgust, anger, and anticipation—into a circular structure, pairing them as opposites to form an emotion wheel. Each primary emotion is further divided into high- and low-intensity counterparts, and secondary emotions emerge from blending two primary emotions. His representation of the emotion wheel is illustrated in Figure 2.3. Notably, his concept of

emotion intensities aligns with dimensional models, where arousal serves as a key dimension for defining emotional intensity.

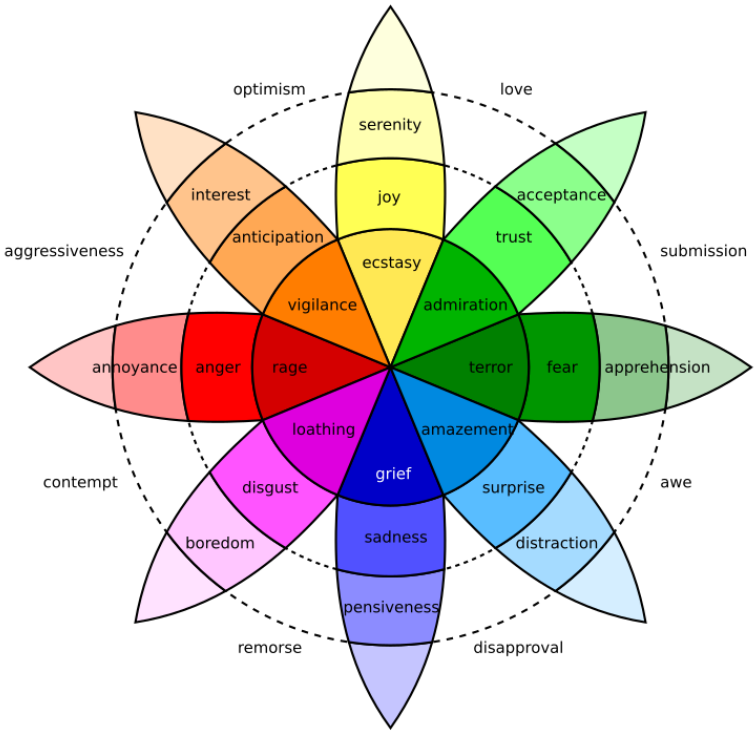


Figure 2.3: Emotion wheel of Plutchik (1988). Author/Copyright holder: Machine Elf 1735. Copyright terms and licence: Public Domain.

2.2.2 Dimensional emotion representation

Dimensional models of emotion represent affect as points in a continuous space instead of fixed categories. These models use numerical values along different axes to describe emotions, allowing for a more flexible and detailed representation of emotional states. Russell (1980) famously introduced the circumplex model of affect, which maps emotions onto a two-dimensional circular plane. The horizontal axis (valence) indicates how pleasant or unpleasant an emotion is (e.g., joy is positive, while sadness is negative). The vertical axis (arousal) represents the level of activation or energy (e.g., anger is positive, while calmness is negative).

Due to its simplicity in representing emotions with just two continuous values, the circumplex model is widely applied in emotion analysis tasks, including emotion prediction from images (Savchenko, 2023), physiological signals (Wiem & Lachiri, 2017), and music (J. Kim et al., 2011). Figure 2.4 illustrates the circumplex model, while showing how categorical emotions are distributed within its quadrants.

Having covered the key data modalities—MIDI and emotion—we now introduce the architecture central to our work: the transformer.

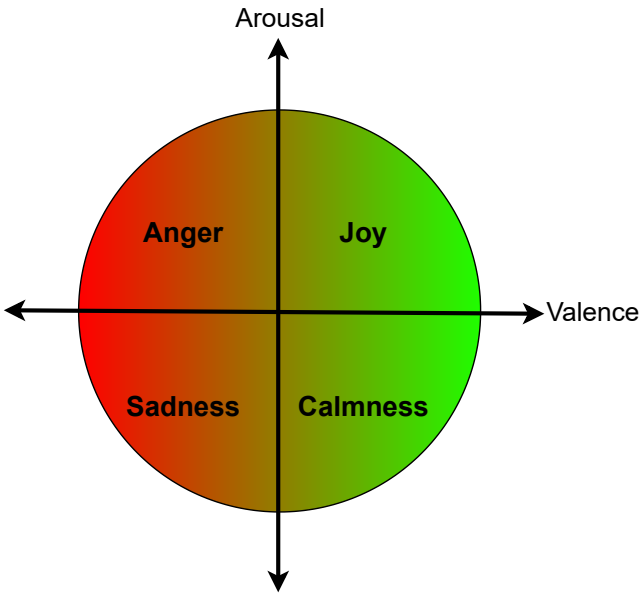


Figure 2.4: Circumplex model of emotions (Russell, 1980).

2.3 Transformers

In this thesis, we tackle several sequence processing tasks, which we describe in detail in later chapters. MIDI is treated as a sequence of notes, and transformers are used to model and generate these sequences. In video classification, features extracted from individual frames form a sequence that can be processed by transformers to predict emotions or cinematic genres. We also work with raw audio, which is often too lengthy to handle in a single pass. To manage this, we split the audio into chunks, extract features from each chunk, and then process the resulting sequence using transformers. Similarly, in text-based tasks, sequences of words or subwords are fed into transformers to perform emotion classification.

Transformers have revolutionized deep learning, particularly in natural language processing (NLP). They form the backbone of all large language models, including ChatGPT[1], Llama[2], and Gemini[3]. Unlike recurrent models, transformers process entire sequences in parallel, eliminating the need for recurrence. By leveraging attention mechanisms and parallel computation, transformers have set new performance benchmarks in NLP tasks such as machine translation (Costa-jussà et al., 2022), text generation (Iqbal & Qureshi, 2022), and question answering (Zaib et al., 2022). However, their superiority extends beyond NLP. Modern architectures for automatic speech recognition (Radford et al., 2023), image processing (Dosovitskiy et al., 2021), and MIDI generation (C.-Z. A. Huang et al., 2019) also rely on transformers.

We now explain the theory behind transformers, beginning with the block diagram shown in

[1] https://chatgpt.com
[2] https://llama.com
[3] https://gemini.google.com

Figure 2.5. The full architecture consists of an encoder (left) and a decoder (right). This encoder-decoder setup is commonly used in tasks like machine translation, where the encoder processes the source sequence and the decoder generates the target sequence. In purely generative tasks, such as text or MIDI generation, only the decoder is used. In this configuration, the target sequence is a shifted version of the input sequence, meaning that for an input token at position n, the target token is at position $n+1$. In simpler terms, given an input sequence, the model predicts the next token.

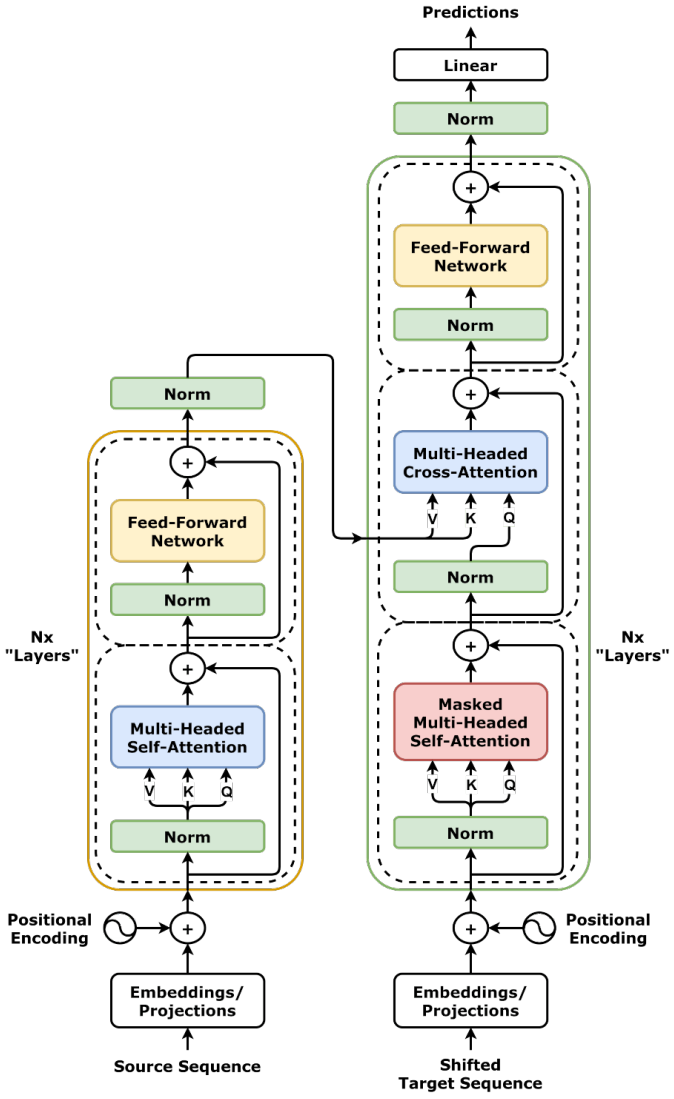


Figure 2.5: The transformer model, with the encoder on the left and the decoder on the right. - https://github.com/dvgodoy/dl-visuals/?tab=readme-ov-file, CC BY 4.0

Tokens are discrete elements of a sequence. In MIDI processing, tokens correspond to individual MIDI messages, such as triggering a note with a specific pitch. In text processing, tokens typically represent words, though subwords are often used to capture semantic relationships between word variations (Sennrich et al., 2016b). This prevents redundant tokenization of words

with shared roots, improving efficiency.

After each token is extracted, unique tokens define the *vocabulary*. Each unique token can be represented with a numerical index within the vocabulary, allowing the input sequence to be represented as a sequence of discrete numbers. Embedding refers to the process of projecting these discrete numbers into vectors with continuous values. The embedding layer is simply a lookup table, defined as a weight matrix with size $V \times D$, where V is the vocabulary length and D is the model dimensionality. The embedding layer maps each token's index to the corresponding row in this matrix, extracting a vector of length D. These vectors are then concatenated along the sequence dimension. For an input sequence of length S, the embedding layer produces a matrix of continuous values with size $S \times D$.

Positional encoding Transformers, unlike Recurrent Neural Networks (RNNs) (Rumelhart et al., 1986), do not process sequences sequentially. Instead, they process the entire input sequence in parallel, significantly improving efficiency. However, this parallelization introduces a key challenge: transformers lack an inherent sense of word order because self-attention treats all input tokens simultaneously.

The example below shows the importance of positional order. Without positional information, the following two sentences would be interpreted as identical:

1- "I only said that she could help him."

2- "I said that only she could help him."

These two sentences are different because of the *position* of the word "only," which changes the emphasis and meaning. In the first sentence, "only" modifies the verb "said." This means that the speaker's action was limited to just saying that she could help him, and not doing anything more. It suggests that the speaker didn't imply anything else beyond saying that she could help. In the second sentence, "only" modifies "she." This means that the speaker is emphasizing that she is the exclusive person who can help him. No one else, except for her, can help.

In order to incorporate positional information, transformers utilize *positional encodings*, which can be defined in various ways. The original transformer employs sinusoidal encodings:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/D}}\right) \tag{2.1}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/D}}\right) \tag{2.2}$$

This results in an interleaved pattern of sine and cosine waves, as illustrated in Figure 2.6.

This position information is injected into the model by simple addition to the embedded sequence. More recent models use learned positional encoding instead of fixed sinusoidal waves (B. Wang et al., 2020). This learned encoding is represented as an $S_{max} \times D$ matrix, where S_{max} is the maximum input sequence length.

C.-Z. A. Huang et al. (2019) introduced the *music transformer*, where relative position encoding (Shaw et al., 2018) is intelligently integrated into the attention mechanism. Unlike absolute

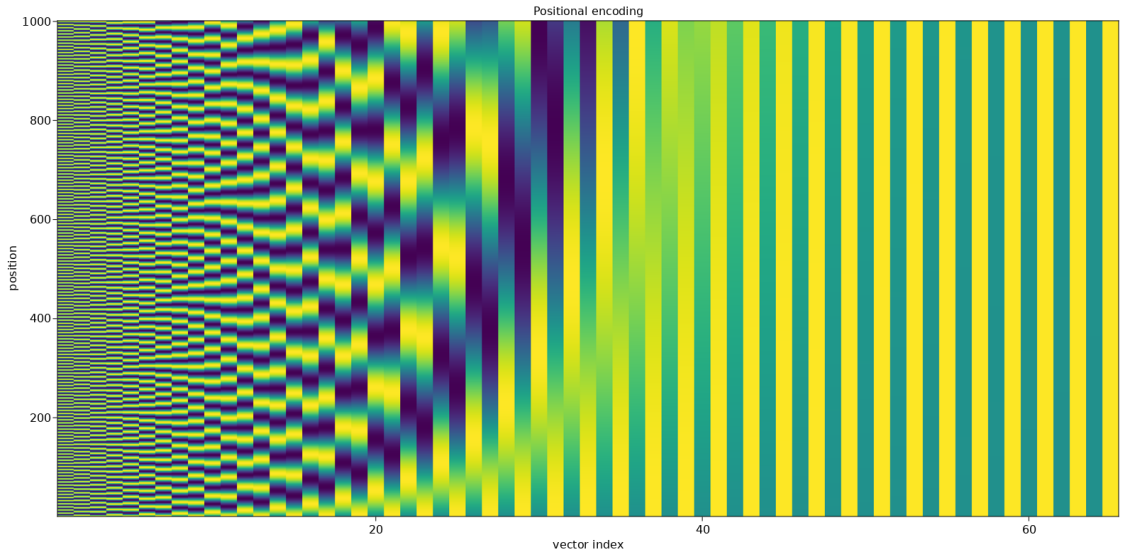


Figure 2.6: Sinusoidal positional encoding. By Cosmia Nebula - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=119948133

positional encoding, relative positions encode the distances between tokens, rather than their absolute positions. In music, rhythm and harmony are defined by how notes interact with one another, not by their fixed time locations. Therefore relative positional encodings are particularly well-suited for music generation, as they reflect the relationships between musical events, rather than their absolute positions in time.

Normalization Normalization of internal activations is crucial for stable training in any neural network. While traditional neural networks typically use batch normalization, which normalizes activations along the batch dimension (Ioffe & Szegedy, 2015), transformers use layer normalization, where activations are normalized across the feature dimension (Ba et al., 2016).

Multi-head attention At the core of the transformer is the attention mechanism, which computes a weighted sum of input values based on their relevance. This enables transformers to process all words in a sentence simultaneously and determine the importance of each word relative to the others.

Consider the example of translating a sentence from a source language to a target language. In self-attention, the model focuses on understanding relationships between words within the target sentence. In cross-attention, the model still focuses on the target sentence, but also considers the influence of the source sentence. While the mechanism behind self- and cross-attention is the same, the key difference lies in the input.

In multi-head attention, the word embeddings are first projected into queries, keys, and values using separate projection matrices. In self-attention, queries, keys, and values are derived from

the same sentence. In contrast, in cross-attention, the source sequence forms the keys and values, while the target sequence forms the queries.

$$Q = X_Q W_Q, \quad K = X_K W_K, \quad V = X_V W_V \tag{2.3}$$

In the above formulas, X_Q, X_K, and X_V represent the input word embeddings for query, key, and values, respectively. W_Q, W_K, and W_V represent the learned projection matrices.

The attention mechanism then calculates the interaction between the keys and queries to assign a weight (importance) to each value.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{D_k}}\right)V \tag{2.4}$$

The query (Q), key (K), and value (V) matrices determine the attention output. The dot product QK^T computes similarity, while $\sqrt{D_k}$, the square-root fo the dimensionality of the keys, scales the values to prevent large gradients. The softmax function converts the scores into probabilities, which weight the values in V.

Attention works by comparing the importance of words to one another through queries (Q), keys (K), and values (V). These are learned representations of the input sequence. The query (Q) represents what we are looking for, such as a word that needs context. The key (K) serves as the reference against which we compare, such as other words in the sequence. The value (V) holds the actual information that is processed, such as the word to which we are assigning importance.

Instead of computing attention once, multi-head attention applies it multiple times in parallel:

$$\text{head}^i = \text{Attention}(W_Q^i, W_K^i, W_V^i) \tag{2.5}$$

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O \tag{2.6}$$

Each attention head (head_i) projects the inputs using different weights. The outputs from all heads are concatenated and linearly transformed by W_O. Using multiple heads allows the model to capture different relationships in parallel.

Feed-forward Network The feed-forward network, represented by the yellow blocks in Figure 2.5 processes each word individually, without considering interactions between them. It typically consists of two linear layers with a rectified linear unit (ReLU) activation in between (Nair & Hinton, 2010). The first linear layer projects the input to a higher dimensionality, while the second linear layer reduces it back to the model's original dimensionality. This expansion in dimensionality enhances the network's representation capacity.

To stabilize training, residual connections are used throughout the model (He et al., 2016). The final linear layer projects the activations back to a dimensionality corresponding to the vocabulary size. The output vector thus represents the probability of each word in the vocabulary.

Having introduced the key concepts, we now turn to a review of related literature on the tasks addressed in this work.

2.4 Sequence processing

Symbolic music can be represented as sequential data, similar to text. Hence, the same models can, in principle, be used for both natural language processing (NLP) and symbolic music processing. One of the oldest neural network architectures for sequence modeling is the recurrent neural network (RNN), where a single input sample (token) is processed at each timestep (Rumelhart et al., 1986). The network is trained by calculating the gradient of the error across each timestep, using the algorithm named *backpropagation through time*.

However, RNNs aren't very successful in modeling long sequences because, as the sequence grows longer, the backpropagated gradients can approach zero. This problem is known as the *vanishing gradient problem*. Long short-term memory (LSTM) networks alleviate this issue by using specialized gates (Hochreiter & Schmidhuber, 1997). A similar variant, the gated recurrent unit (GRU), can achieve performance similar to LSTM but with a simpler architecture and fewer parameters (Cho, van Merrienboer, et al., 2014b).

Despite these improvements, even LSTMs and GRUs struggle to process long sequences efficiently and tend to "forget" earlier input samples as the sequence grows. The attention mechanism addresses this issue by explicitly modeling dependencies between *all* pairs of input samples (Bahdanau et al., 2015). Finally, the transformer model has set the current state-of-the-art in sequence processing by incorporating the attention mechanism in a multi-headed and multi-layered architecture (Vaswani et al., 2017).

The original transformer implementation consisted of an encoder and a decoder network, and it was tested on the task of machine translation. Encoder-decoder architectures are commonly used for machine translation, where the encoder processes the source text and the decoder generates the output (Cho, van Merrienboer, et al., 2014a). Since language modeling involves generating text from scratch, it can be seen as analogous to music generation, meaning both tasks can be categorized under the task of *sequence generation*.

In sequence generation, there are no separate source and target sequences. As a result, state-of-the-art language models consist solely of a decoder (Brown et al., 2020). Sequence-generating neural networks are trained with input and target sequences from the same domain. Specifically, the target sequence is a shifted version of the input sequence, so for each input token, the network predicts the next token.

During the training of a sequence generation model, the training loss is calculated based on the prediction performance of all tokens, even though the model only considers the previous tokens. To ensure that the model does not attend to future tokens, a triangular mask is added to the softmax output of the attention module. An example of a triangular mask for a sequence length of 5 is shown below.

$$mask = \begin{bmatrix} 0 & -\infty & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty & -\infty \\ 0 & 0 & 0 & -\infty & -\infty \\ 0 & 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

This ensures that the attention outputs related to future tokens are masked. For example, while the model is predicting the 3rd token, it will only use the 1st and 2nd tokens. This generation task can be viewed as a classification task for all tokens. The most commonly used loss function for this task is cross-entropy loss:

$$\mathscr{L} = -\sum_{s=1}^{S}\sum_{v=1}^{V} y_{s,i} \log \hat{y}_{s,i} \tag{2.7}$$

This loss function measures the difference between the predicted token probabilities and the true labels. The index s represents the position in the sequence, ranging from 1 to S, where S is the total number of tokens. The index i corresponds to the vocabulary word indices, ranging from 1 to V, where V is the vocabulary size. The term $y_{s,v}$ is a one-hot encoded label, meaning it is 1 if the correct word at position s is word v, and 0 otherwise. The term $\hat{y}_{s,v}$ is the predicted probability of word i at position s, which is obtained from the softmax output of the transformer model. The logarithm $\log \hat{y}_{s,v}$ ensures that incorrect predictions are heavily penalized, as lower predicted probabilities lead to larger negative values. Since only one word is correct at each position, only one term in the summation contributes to the loss at each timestep. The model optimizes this function by minimizing $\mathscr{L}$, encouraging higher probabilities for the correct words while reducing the likelihood of incorrect predictions.

Sequence generation models operate autoregressively during inference, meaning that each new token is generated one at a time. To generate each token, a forward pass is performed. After generating a token, it is added to the sequence, and this updated sequence is then fed back into the model to predict the next token. For instance, if we have already generated a sequence consisting of tokens 1 and 2, we use this sequence as input to predict token 3. Once token 3 is generated, it is appended to the sequence, forming the sequence [1, 2, 3], which is then used to predict token 4, and so on. Often, a special `START` token is used as the first token to initiate inference from scratch. Alternatively, inference can be performed using a predefined sequence, known as a *primer* or *priming sequence*. In this case, the model continues from the sequence provided by the user, generating tokens that extend the given sequence.

In contrast to generative tasks, classification tasks predict a single set of probabilities for the entire sequence, rather than for each token. To facilitate this, the input sequence is prepended with a special token called the `CLS` (classification) token, and only the output corresponding to this token is considered. In classification tasks, no triangular mask is applied, so each token can attend to every other token. As a result, the output corresponding to the `CLS` token utilizes the entire sequence.

In the following sections, we discuss sequence classification and sequence generation tasks that are relevant to our work. We begin with a review of text emotion classification, followed by conditional sequence generation, and specifically conditional music generation.

2.4.1 Text emotion classification

Emotion classification from text—or sentiment analysis, as used interchangeably in the machine learning literature—allows us to automatically identify and/or quantify the emotion expressed in a piece of text, such as a review, social media post, or customer feedback (B. Liu & Zhang, 2012). Identifying the underlying emotion in text is useful in various fields such as customer service (M. Hu & Liu, 2004), finance (Nguyen et al., 2015), politics (Iyyer et al., 2014), and entertainment (J. Almeida et al., 2021).

We now present the most recent and comprehensive text emotion classification datasets, along with the methods.

2.4.1.1 Datasets

The SemEval (Semantic Evaluation) workshops regularly host a variety of challenges focused on semantic text analysis (Ojha et al., 2024). Several of these challenges target text-based emotion classification. Task 11 of the 2025 edition, titled "Bridging the Gap in Text-Based Emotion Detection", compiles samples from diverse sources including social media, speeches, literature, and news (Muhammad et al., 2025). The dataset includes over 100,000 samples across more than 30 languages, all manually labeled with Ekman's six basic emotions—joy, sadness, anger, fear, surprise, and disgust—along with a neutral category (Ekman, 1971). Of these, 5,651 samples are in English. In the 2019 edition, Task 3—"EmoContext: Contextual Emotion Detection in Text"—challenges participants to determine the emotion of the final utterance in a short dialogue (Chatterjee et al., 2019). This dataset contains 38,424 samples labeled with four categories: happy, sad, angry, and others. Task 1 of the 2018 edition, titled "Affect in Tweets", focuses on emotional content in Twitter data[4] (Mohammad et al., 2018). It includes multiple subtasks: emotion intensity regression, ordinal classification of intensity, valence regression, valence ordinal classification, and emotion classification. The challenge supports three languages—English, Spanish, and Arabic—with 10,983 English samples specifically available for the emotion classification task.

The EmoBank dataset consists of 10,062 English sentences collected from sources such as news articles, blogs, fiction, and travel guides. These samples are annotated through crowdsourcing using a dimensional emotion representation scheme (Buechel & Hahn, 2017). The Multimodal EmotionLines Dataset (MELD) is designed for emotion recognition in multi-party conversations (Poria et al., 2019). It includes text, video, and audio data. The text component features 13,000 English utterances labeled with Ekman's six basic emotions—joy, sadness, anger, fear, surprise, and disgust—along with a neutral category (Ekman, 1971). The Cleaned Balanced Emotional Tweets (CBET) dataset provides 81,163 tweets annotated with nine emotion labels: joy, anger,

[4] https://twitter.com

sorrow, love, contempt, surprise, fear, guilt, and thankfulness (Shahraki, 2015). The GoEmotions dataset contains 58,000 English text samples sourced from Reddit[5] comments (Demszky et al., 2020). It uses a broad range of 27 categorical emotion labels, including admiration, amusement, approval, annoyance, anger, curiosity, confusion, caring, desire, disgust, disapproval, disappointment, embarrassment, excitement, fear, joy, grief, gratitude, love, sadness, nervousness, optimism, pride, remorse, relief, realization, and surprise.

Table 2.1 provides a summary of the text emotion datasets mentioned above. It outlines the source or context of each dataset, the number of English-language samples, and the type of emotion representation used.

Table 2.1: Text emotion classification datasets

Dataset	Context	Sample size	Emotion representation
CBET (Shahraki, 2015)	Tweets	81,163	Categorical
EmoBank (Buechel & Hahn, 2017)	News, blogs, fiction, travel guides	10,062	Dimensional
SemEval-2018 (Mohammad et al., 2018)	Tweets	10,983	Categorical
SemEval-2019 (Chatterjee et al., 2019)	Dialogues	38,424	Categorical
MELD (Poria et al., 2019)	Dialogues	13,000	Categorical
GoEmotions (Demszky et al., 2020)	Reddit comments	58,000	Categorical
SemEval-2025 (Muhammad et al., 2025)	Social media, speeches, literature, news	5651	Categorical

2.4.1.2 Methods

Machine learning methods have significantly advanced the state of the art in text emotion classification for the past two decades. However, the earliest works in this field relied on hand-crafted features, such as frequently used n-grams (Pang et al., 2002), or adjectives and adverbs that are associated with particular emotions (Whitelaw et al., 2005). Nonetheless, the advent of deep learning has made it computationally feasible to process raw inputs without extracting features manually, leading to better performance (Krizhevsky et al., 2012). Recurrent Neural Networks and their improved variants such as Long Short-Term Memory were initially used (Rumelhart et al., 1986) but were later replaced by the transformer model (Vaswani et al., 2017), which is the current state of the art in text classification tasks. The 2025 SemEval task on text emotion classification, titled "Bridging the Gap in Text-Based Emotion Detection", highlights that current state-of-the-art approaches rely on pretrained transformers with fine-tuning, or large language models (LLMs)

[5] https://www.reddit.com

enhanced through instruction tuning, adapter modules, or data augmentation techniques (Muhammad et al., 2025).

Fine-tuning pretrained transformers for specific tasks leverages the knowledge acquired during their initial training phase. The GPT (Generative Pretraining Transformer) model, for instance, is a large transformer pretrained on next-token prediction and later fine-tuned for various NLP tasks, achieving state-of-the-art performance (Radford et al., 2018). BERT (Bidirectional Encoder Representations from Transformers) enhanced this approach by introducing masked token prediction during pretraining, which allowed the model to learn bidirectional context (Devlin et al., 2019). Building on BERT, the RoBERTa (Robustly Optimized BERT Approach) model further improved performance by employing extensive hyperparameter tuning, training on more data, using larger batch sizes, and extending the pretraining duration (Y. Liu et al., 2019).

Instruction-tuning involves the supervised fine-tuning of large language models (LLMs) using instructions, typically in the form of prompt-output pairs (S. Zhang et al., 2023). Since fine-tuning an entire LLM can be resource-intensive, many approaches reduce the number of trainable parameters by using adapters (Houlsby et al., 2019). Adapters are small modules inserted within a neural network, enabling fine-tuning of the model while keeping the large network's weights frozen. This allows the adapter modules to be trained while maintaining computational efficiency. Advanced adapters, like Low-Rank Adapters (LoRA), use low-rank matrix decomposition to represent weight updates with fewer parameters than the original weight matrix (E. J. Hu et al., 2022). Leon et al. (2025) utilized adapters for cross-lingual text emotion classification, leveraging datasets from multiple languages. Ranjbar and Baghbani (2025) employed LLMs for data augmentation in fine-tuning pretrained transformers. They used the LLaMa-3 model[6] to generate explanations for text in their training dataset, enhancing the performance of a pretrained RoBERTa model on text emotion classification tasks.

We continue by providing background on sequence processing, focusing on conditional sequence generation.

2.4.2 Conditional generic sequence generation

Although it is a loosely used term, *conditioning* refers to controlling a model's output by providing auxiliary inputs, i.e., conditions. These conditions can belong to the same domain as the input and target, enabling training with unlabeled data. Alternatively, conditions can belong to different domains, such as the labels of a labeled dataset.

Even the earliest neural networks for natural language processing utilized conditioning. Mikolov and Zweig (2012) developed a language model conditioned on factors such as topic and genre, where a conditioning vector was created using a linear layer and concatenated with the hidden state of the recurrent neural network (RNN). Sennrich et al. (2016a) developed an encoder-decoder RNN model for translation from English to German, conditioned on politeness. They used control tokens to specify whether the user preferred a formal or informal translation.

[6] https://llama.com

The Conditional Transformer Language (CTRL) model feeds control tokens that denote domain, style, topics, and more into a large transformer, achieving state-of-the-art results in conditional language modeling (Keskar et al., 2019). Krishna et al. (2020) performed style transfer by generating paraphrases and demonstrated that training separate models for each style outperforms training a single model that uses style-specific control tokens.

Sheng et al. (2020) identified triggers—subsequences that generate biased text when used as inputs—and employed them as primers to induce or balance bias in language modeling. Smith et al. (2020) investigated controlling the style of dialogue generation by comparing three methods: retrieve-and-refine (Weston et al., 2018), inference-time iterative refinement (Dathathri et al., 2020), and conditional generation using control tokens (Keskar et al., 2019). They showed that conditional generation with control tokens outperformed the other methods.

Most works in the literature use categorical variables, such as control tokens, to guide language modeling. In contrast, image captioning can be framed as a text generation task based on images, which are non-categorical variables. In this context, the input image is typically processed by a convolutional neural network (CNN), and the resulting features are used to condition a separate language model. Earlier works used recurrent neural networks (RNNs) as the language model for this task (Vinyals et al., 2017), but state-of-the-art models have replaced RNNs with transformers (X. Zhu et al., 2018). X. Zhu et al. (2018) compared different conditioning methods and observed similar performance across them. These methods include feeding the spatial image features into the cross-attention layer of the decoder (K. Xu et al., 2015), combining the image features with each word embedding, and feeding the image features before the word embeddings (Vinyals et al., 2017).

Next, we move on to the task most closely related to our overall work: conditional music generation.

2.4.3 Conditional symbolic music generation

While we maintain our focus on the conditional generation of symbolic music, it is essential to understand non-conditional generation of symbolic music as well. We proceed by presenting the relevant datasets and methods.

2.4.3.1 Datasets

The largest symbolic music dataset is the Lakh MIDI Dataset (LMD) (Raffel, 2016), which includes 176,581 unlabeled, multi-instrument MIDI files. Of these, 45,129 files are matched to 31,034 tracks in the Million Song Dataset (MSD) (Bertin-Mahieux et al., 2011). The mismatch in numbers arises because multiple MIDI files can correspond to the same MSD track, and a single MIDI file can also map to multiple tracks.

The MAESTRO dataset (MIDI and Audio Edited for Synchronous TRacks and Organization) is a large, high-quality dataset designed for music research, especially in areas like transcription, generation, and synthesis (Hawthorne et al., 2019). It includes over 200 hours of professional

piano performances, with precisely aligned audio and MIDI recordings. The MIDI data captures not just the notes and timing but also note velocities, which represent the intensity or dynamics of each keystroke—crucial for modeling expressive piano performance.

Zhuo et al. (2023) introduced the Symbolic Music Videos (SymMV) dataset, which contains video-MIDI pairs. They sourced MIDI data from piano tutorial videos, automatically transcribing audio using the Onsets and Frames model (Hawthorne et al., 2018), resulting in 1,140 samples. They also provide low-level features, such as color histograms and RGB frame differences for motion, as well as high-level CLIP features extracted from the videos.

To the best of our knowledge, there are only four publicly available symbolic music datasets with emotion labels, though their sample sizes are quite small. The VGMIDI (Video Game MIDI) dataset consists of 204 video game soundtracks played on piano, with continuous-valued labels for valence and arousal annotated by 30 human subjects (Ferreira & Whitehead, 2019). Panda et al. (2013) created the MIREX-like dataset with discrete emotion labels sourced from the AllMusic database[7]. The dataset primarily consists of audio recordings, but 193 of the samples are also available as MIDI files, featuring a varying number of instruments. The EMOPIA (EMOtion PIAno) dataset consists of paired audio and MIDI clips featuring solo piano, extracted from 387 songs (Hung et al., 2021). Emotional labels were manually annotated by the authors using discrete categories based on the four quadrants of the two-dimensional circumplex model of affect (Russell, 1980). Table 2.2 presents the MIDI datasets, the total durations in hours, instrumentations and the types of emotion labels.

Unfortunately, due to their small sizes, these datasets are insufficient for training deep neural networks with millions of parameters, as they tend to cause overfitting and poor generalization. Overall, there are no large-scale, multi-instrument MIDI datasets annotated with emotion labels.

Table 2.2: MIDI datasets

Dataset	Duration (hours)	Instrumentation	Emotion label
MIREX-like (Panda et al., 2013)	11.6	Multi-instrument	Categorical
LMD (Raffel, 2016)	9382.1	Multi-instrument	None
MAESTRO (Hawthorne et al., 2019)	200.0	Piano-only	None
VGMIDI (Ferreira & Whitehead, 2019)	6.4	Piano-only	Dimensional
EMOPIA (Hung et al., 2021)	11.0	Piano-only	Categorical
SymMV (Zhuo et al., 2023)	76.5	Piano-only	None

[7] https://allmusic.com

2.4.3.2 Methods

Early works using neural networks for music generation employed recurrent neural networks (RNNs) (Eck & Schmidhuber, 2002). However, the recent advent of the transformer model has enabled the modeling of much longer dependencies. The music transformer builds upon the transformer model by incorporating relative positional information, providing a better representation of the relationships between notes (C.-Z. A. Huang et al., 2019).

The majority of existing literature on symbolic music generation relies on a non-conditional approach. These methods are trained on raw MIDI data without explicit labels, enabling the generation of new music similar to the examples in the training dataset (C.-Z. A. Huang et al., 2019). However, some approaches leverage low-level features within the data to generate music in a conditional manner (C.-Z. A. Huang et al., 2017). For example, they may use short melodies, chords, or single-instrument tracks as input to generate corresponding melodies. Using short melodies as input is particularly practical because they are directly fed into the model, requiring no changes to the architecture. In sequence processing, these short input sequences are referred to as *primers*. The model then predicts the melody that follows the primer melody. In this case, both the condition and the target belong to the same domain, allowing the model to be trained using unlabeled data.

Other symbolic music generation tasks that utilize same-domain conditioning include accompaniment generation (Dong et al., 2018; C.-Z. A. Huang et al., 2017), interpolation (Roberts et al., 2018), inpainting (K. Chen et al., 2020; Ens & Pasquier, 2020), and style transfer (Choi et al., 2020; Z. Wang et al., 2020). MidiNet can generate melodies conditioned on chords by training on a private dataset that includes chord information (L.-C. Yang et al., 2017). OpenAI's MuseNet is trained on a combination of datasets, and generation can be conditioned on specific artist names, genres, or styles using primer control tokens (Payne, 2019).

It is also possible to use low-level symbolic music features for conditioning (Guo et al., 2020; Park & Kim, 2000a; R. Yang et al., 2019). Features such as tempo, note density, pitch range, and tonal tension can be automatically calculated, eliminating the need for a labeled dataset. H. H. Tan and Herremans (2020) aimed to compensate for the small size of the labeled VGMIDI dataset by augmenting it with the unlabeled MAESTRO (MIDI and Audio Edited for Synchronous TRacks and Organization) dataset (Hawthorne et al., 2019)—detailed in the next section. They used low-level rhythm and note density features to infer the high-level arousal feature. Similarly, J. Zhu et al. (2024) incorporated chord information using additional tokens and further enhanced output quality through the use of Generative Adversarial Networks (Goodfellow et al., 2020).

Emotion-based music generation, or equivalently Affective Algorithmic Composition (AAC) methods focus on the automatic composition of music based on specific emotions (Williams, Kirke, Miranda, et al., 2015). Use cases for AAC include composing soundtracks for videos and video games (Williams, Kirke, Eaton, et al., 2015), neurofeedback training for medical applications (Pratt et al., 1995), and developing brain-computer music interfaces (Miranda et al., 2011).

Although the relationship between music and emotion is well-studied (Juslin & Sloboda, 2001), choosing the "optimal" emotion model to explore this relationship remains a debated topic (Eerola & Vuoskoski, 2011). Existing AAC work primarily employs the two main categories of emotion models: categorical and dimensional (Williams, Kirke, Miranda, et al., 2015).

As discussed in Section 2.2.1, categorical emotion models use discrete labels such as happy, sad, angry, and surprised (Ekman, 1975). However, studies on dimensional approaches argue that categorical models fail to capture the complexities and subtleties of human emotions. Instead, they propose using continuous-valued coordinates to represent emotions in a low-dimensional space (Gunes et al., 2011).

Early works on AAC used various melodic, harmonic, and rhythmic features to target specific emotions (see the overview of Williams, Kirke, Miranda, et al. (2015)). However, this is an indirect approach, and the correspondence between emotions and musical features is only approximate (Berg & Wingstedt, 2005; Wingstedt et al., 2005). The advent of deep learning allowed for the use of complex models trained on large labeled datasets, utilizing the emotion labels directly and eliminating the need for intermediate features (Krizhevsky et al., 2012).

More recent AAC models are trained in an end-to-end fashion on datasets containing symbolic music and emotion labels. The creators of the VGMIDI dataset developed a method for symbolic music generation conditioned on emotion (Ferreira & Whitehead, 2019). Using a genetic algorithm, they fine-tuned the weights of a pretrained LSTM, separately for positive and negative valence conditions, resulting in two models.

Both K. Zhao et al. (2019) and Hung et al. (2021) generated symbolic music conditioned on four categorical emotions, each corresponding to one of the four quadrants of the valence-arousal plane. K. Zhao et al. (2019) labeled the classical piano music obtained from piano-midi.de, using categorical emotion labels and trained a biaxial LSTM (Johnson, 2017) on this labeled dataset. Hung et al. (2021), the creators of the EMOPIA dataset, trained a transformer model conditioned with control tokens (Keskar et al., 2019). However, these works were limited by the small number of categorical emotion labels they could use, likely due to the small sizes of their training datasets.

We reserve Section 2.6 to discuss *video-based symbolic music generation*, as this task requires more complex models with video analysis modules. We now proceed by discussing the task of video analysis.

2.5 Video analysis

Video-based music generation models typically consist of a video analysis module followed by a music generator (Di et al., 2021; Kang et al., 2024). As explained in Chapter 1 our work involves both low-level video analysis, specifically scene cut detection, and high-level video analysis, namely semantic video classification.

Scene cut detection is built into standard video processing libraries, such as FFMpeg[8]. FFmpeg's algorithm computes the average pixel difference between consecutive frames and marks a

[8] https://ffmpeg.com

scene cut when this difference exceeds a specified threshold. As scene cut detection is a solved problem, we center our discussion on semantic video classification, with a particular focus on emotion and genre classification.

Unlike the classification of other signals, video classification is a *multimodal* task. Video classification models utilize both video frames—capturing scenes, objects, actions, faces, and text—and audio, which includes speech, music, effects, and audio events. Additionally, since videos are sequences, both short- and long-term dependencies must be considered. Furthermore, the nature of the video content significantly influences which modalities are most relevant. For instance, when classifying emotions based on facial expressions, visual information from video frames is essential. In contrast, for detecting emotions in social interactions, analyzing speech becomes equally or even more important.

2.5.1 Video emotion classification

Video emotion classification is the task of automatically detecting and categorizing emotions expressed in videos using computational methods. Video classification models take video frames as input, and often include audio as well, producing output in the form of probabilities for predefined emotion categories or continuous values such as valence and arousal. While our work focuses on the classification of arbitrary videos, we also list the following video emotion datasets that include both arbitrary and specific video types.

2.5.1.1 Datasets

Gao et al. (2021) present the Pairwise Emotional Relationship Recognition (PERR) dataset which includes 31,182 videos from TV drama series, focusing on social interactions between humans. The Context Aware Emotion Recognition (CAER) dataset similarly collects 13,201 clips from 79 TV series, but it specifically selects clips containing human faces (Lee et al., 2019). Pandeya and Lee (2021) label the emotions in 3,438 music videos, creating the Music Video Emotion Dataset (MVED).

There are limited datasets for the classification of arbitrary videos, i.e., those that include a diverse set of videos. The VideoEmotion-8 dataset consists of 1,001 user-generated videos collected from YouTube and Flickr, labeled with 8 categorical emotions: anger, anticipation, disgust, fear, joy, sadness, surprise, and trust (Jiang et al., 2014). The Ekman-6 dataset (B. Xu et al., 2018) is an extended version of VideoEmotion-8, containing 1,637 videos labeled with the six basic emotions defined by Ekman (1971): anger, disgust, fear, joy, sadness, and surprise. Table 2.3 summarizes the video emotion datasets, all of which use categorical emotion representations.

We now proceed to discuss methods for video emotion classification.

2.5.1.2 Methods

All modern video classification models rely on deep neural networks. B. Xu et al. (2018) employed cross-domain knowledge transfer, particularly from image emotion classification to video emotion

Table 2.3: Video emotion datasets.

Dataset	Context	Sample size
VideoEmotion-8 (Jiang et al., 2014)	User-generated	1,001
Ekman-6 (B. Xu et al., 2018)	User-generated	1,637
CAER (Lee et al., 2019)	Human faces	13,201
PERR (Gao et al., 2021)	Social interactions	31,182
MVED (Pandeya & Lee, 2021)	Music videos	3,438

classification. C. Chen et al. (2016) first classify scenes, objects, and events in the video and then fuse this information to classify the emotion. S. Zhao et al. (2020) used 3D convolutional neural networks to process video frames and 2D convolutional neural networks to process audio spectrograms. They also used modified loss functions that apply a higher penalty for incorrect classifications when the prediction and the target share similar emotional polarities.

Z. Zhang et al. (2023) introduced inter- and intra-attention modules to analyze dependencies within and between video frames and audio. Based on the softmax outputs from the attention modules, they identify dominant features and then erase them, encouraging the model to focus on secondary information.

Due to the long length of video frame sequences, many methods extract and operate on *keyframes*. A straightforward way to extract keyframes is by identifying scene boundaries, but Wei et al. (2021) further filter these based on their emotional saliency. They calculate emotional saliency by feeding frames into an image emotion classifier and using the probability of the highest class. Essentially, they select video frames that produce more pronounced emotion classification results, and use those frames in their final analysis. Yi et al. (2024) applied data augmentation techniques for video emotion classification by blending frames through linear interpolation or cutting and pasting different sections. A majority of the aforementioned methods rely on data-oriented approaches, such as data selection (e.g., identifying key frames) or data augmentation (e.g., generating synthetic frames).

We now continue our discussion of semantic video classification by turning to an alternative task: cinematic trailer genre classification. Thanks to online film databases like IMDb and video platforms such as YouTube, cinematic trailer datasets tend to offer larger sample sizes and longer video durations. This abundance of data allows researchers to shift focus from data-oriented strategies to model-oriented approaches.

2.5.2 Trailer genre classification

In trailer genre classification, models process cinematic trailers to predict the movie's genres. This is usually a multi-label classification task since movies are usually associated with multiple genres. We continue our discussion with cinematic datasets and trailer genre classification methods.

2.5.2.1 Datasets

MovieLens is one of the earliest cinematic datasets, containing movie ratings and metadata such as genre and title (Harper & Konstan, 2016). While the latest version includes 86k movies, it does not provide direct links to the movie trailers. These trailers can only be obtained by crawling the Internet Movie Database (IMDb) website[9], which conflicts with their terms of use.

The LMTD (Large Movie Trailer Dataset) is a collection of features and metadata from 3,500 movie trailers, although the project is currently discontinued and the data is no longer available (Simoes et al., 2016). The MM-IMDb (Multimodal IMDb) dataset merges movie posters with metadata from MovieLens, resulting in 26k movies with posters, plots, genres, and other metadata, but without trailers or videos (Ovalle et al., 2017).

The MMTF-14K (Multimodal Movie Trailer Features dataset) provides multimodal features extracted from 14k movie trailers along with metadata such as user reviews and genre (Deldjoo et al., 2018). MovieScope is a comprehensive dataset for multi-modal movie analysis, including data such as movie trailers, posters, plot synopses, user reviews, and visual-auditory features, covering 5k distinct movies (Cascante-Bonilla et al., 2019).

The Condensed Movies dataset includes full individual scenes, rather than trailers, along with plot details and character information, from 4k movies (Bain et al., 2020). MovieNet is a large-scale, holistic dataset providing movie, trailer, poster, subtitle, plot, tags, and genre metadata (Q. Huang et al., 2020). While it contains metadata for 375k movies, trailers are available via YouTube links for 33k movies, making MovieNet the largest cinematic dataset in terms of labeled trailers.

In the context of trailer genre classification, earlier datasets either do not include trailers or require web scraping, which may conflict with the terms of use of online platforms. More recent efforts provide YouTube links to trailers associated with the movies in their datasets, but these are still limited in size. Among available options, MovieNet stands out as the largest dataset offering direct access to movie trailers, making it the most suitable candidate for training DNNs for trailer genre classification.

2.5.2.2 Methods

One of the earliest works on movie genre classification used scene detectors to obtain keyframes and extracted hand-crafted visual features, such as roughness, ruggedness, and openness, from a privately collected dataset (H. Zhou et al., 2010). Genre classification was achieved by comparing the distance between the feature vectors from the training and testing sets.

One of the first works to use neural networks for movie genre classification also leveraged visual pretrained features (Wehrmann & Barros, 2017), using the LMTD dataset (Simoes et al., 2016). They utilized pretrained features from classification models trained on the ImageNet (Deng et al., 2009) and Places365 (B. Zhou, Lapedriza, et al., 2018) datasets, along with audio spectrograms. The features from individual frames were fused using a convolution-through-time module, which operates as a standard convolutional neural network (CNN) where the kernel length matches

[9] https://www.imdb.com

the input feature length of each keyframe. The kernel traverses features from subsequent frames along the temporal dimension (Wehrmann & Barros, 2017). The kernel length along the temporal dimension is 3, meaning the model exploits the correspondence between only 3 frames. Additionally, since the input and output of the CNN have varying numbers of frames, the model takes the maximum values along the temporal dimension to create a fixed-size vector for the final classification layer. This approach inevitably leads to a loss of information. Another work explored the correspondence between facial emotions and cinematic genres by first extracting human faces from trailer videos, classifying their emotions, and then mapping these emotions to cinematic genres (Yadav & Vishwakarma, 2020).

The work presenting the MovieNet dataset also introduces a model for movie genre classification, along with results from other state-of-the-art video classification models (Q. Huang et al., 2020). While the model they introduce surpasses the performance of existing models, it only uses 8 clips, each containing 3 frames from the entire trailer. During inference, predictions are made for each individual clip, and these are averaged to generate the final prediction. The MovieCLIP model first trains a scene classification model and then uses its final activations as input to a genre classification model, specifically using the Moviescope dataset (Bose et al., 2023).

Some recent works have used transformers for movie genre classification. Rodriguez Bribiesca et al. (2021) used transformers to individually process pretrained frame features, then fused the resulting activations with metadata and movie posters. Miyazawa et al. (2022) and Ak et al. (2023) used pretrained transformers to process movie posters and plots, excluding any use of video data, for genre classification on the MM-IMDb dataset (Ovalle et al., 2017).

All of the aforementioned methods place limitations on input data—either by restricting the number of input frames to a fixed, small set or by relying on pretrained frame features derived from standard image classification models such as VGG (Visual Geometry Group) (Simonyan & Zisserman, 2015). The former fails to capture the long-term dependencies present in trailers, while the latter limits the model to features extracted from a single context, overlooking other informative cues such as human faces or on-screen text. These limitations highlight the need for improved methods that leverage a large and variable number of frames and incorporate pretrained features from a diverse range of tasks.

Next, we discuss works that combine video analysis with symbolic music generation, resulting in video-based symbolic music generators.

2.6 Video-based symbolic music generation

Video-based symbolic music generators take an input video and compose a suitable soundtrack in symbolic format. Several studies focus on generating symbolic music for specific types of videos, such as those featuring human movements like dancing or instrumental performances. The Foley Music model (Gan et al., 2020) generates MIDI from videos of musicians by processing body keypoint movements using a Graph Convolutional Network (Kipf & Welling, 2017) and a Transformer (Vaswani et al., 2017). Similarly, Koepke et al. (2020) and Su et al. (2020) use

deep neural networks with residual connections to generate symbolic music from videos of finger movements on piano keyboards.

Due to their specialized nature, these approaches rely on datasets containing video-MIDI pairs, though these datasets typically contain fewer than 1k samples (Koepke et al., 2020; B. Li et al., 2019; Su et al., 2020; H. Zhao et al., 2018). The RhythmicNet model employs a multi-stage process to generate music from dance videos by predicting beats and style, generating a drum track, and subsequently creating multitrack music (Su et al., 2021). We now present works that address the more general task of generating symbolic music for arbitrary videos.

The Controllable Music Transformer (CMT) generates music based on video features such as motion speed, motion saliency, and timing (Di et al., 2021). Our preliminary explorations indicate that these features are temporally dense and continuously influence music generation, which leads to an inconsistent tempo. CMT employs the Lakh Pianoroll Dataset (Dong et al., 2018) and processes music using an extended compound-word representation (Hsiao et al., 2021). In this representation, each token encodes type, beat/bar marking, note strength, note density, instrument, pitch, and duration. While this reduces the sequence length, it significantly increases the input dimensionality. Moreover, note events in CMT are strictly aligned with beat subdivisions, whereas human musicians often introduce expressive timing deviations—an essential element for conveying emotion in musical performance. In a follow-up work to CMT, Zhuo et al. (2023) trained three generation models sequentially: one for generating chord sequences, another for generating the melody, and a third for generating the accompaniment.

The Video2Music model utilizes both low-level video features and high-level emotional conditioning (Kang et al., 2024). The authors compiled the MuVi-Sync dataset, which consists of 748 music videos labeled with musical attributes such as note density, loudness, chords, and key—but does not include symbolic music data. Their encoder-decoder transformer takes low- and high-level video features, along with a user-provided primer chord and key, to generate chord sequences. These sequences are then arpeggiated using fixed patterns to create the final MIDI output. However, the model's reliance on a fixed time grid eliminates the subtle expressive timing found in human performances. Additionally, its use of fixed arpeggiation patterns limits musical diversity, and the requirement for user-defined chord and key inputs restricts accessibility for non-musicians.

Evaluation of video-based music generators is non-trivial. For objective evaluation, most works measure the difference between generated and ground-truth samples using metrics such as Pitch Class Histogram Entropy, Grooving Pattern Similarity, and Structureness Indicator (S.-L. Wu & Yang, 2020), and Number of Statistically-Different Bins (Engel et al., 2019), or functions like contrastive loss (Alayrac et al., 2020). When the dataset contains paired video and MIDI samples, this method is straightforward (Gan et al., 2020; Zhuo et al., 2023). For unpaired video and MIDI samples, objectively evaluating a model that generates MIDI from arbitrary videos is not feasible. Su et al. (2021) evaluate each module in their multi-stage architecture using ground-truth data from the training datasets corresponding to individual stages—namely, beat and style prediction, drum track creation, and ultimately, music generation.

The authors of the Controllable Music Transformer, using unpaired datasets, evaluate output

MIDIs generated unconditionally, without input video (Di et al., 2021). They empirically show that unconditionally generated MIDI samples exhibit metrics closer to the unpaired MIDI dataset compared to video-conditioned output samples. This occurs because constraining the model with video structure causes deviations from intrinsic MIDI structures, and the best way to match a specific MIDI dataset's metrics is to train exclusively on that dataset.

Overall, existing approaches to video-based music generation exhibit several limitations. Some models are designed for narrow contexts, such as dance or instrument performance videos, limiting their generalizability. Others generate only piano music, restricting the expressive and instrumental range. Methods that rely on continuous temporal conditioning often disrupt the natural sense of tempo, while nearly all approaches use a fixed and coarse time grid, failing to capture the nuanced timing of human musical performance.

Evaluation practices also have notable shortcomings. Some studies assess only individual components of multi-stage models without evaluating the full pipeline. Others evaluate MIDI generation in isolation, ignoring its alignment with the video content. Subjective evaluation through user studies remains the most common—and in many cases, the only—practical approach for assessing video-based MIDI generation (Di et al., 2021; Gan et al., 2020; Kang et al., 2024; Su et al., 2021; Zhuo et al., 2023).

We continue discussing audio generation and its challenges, specifically focusing on the task of audio bandwidth extension.

2.7 Audio bandwidth extension

Modern recording techniques provide music signals with extremely high audio quality. By contrast, the listening experience of archive recordings, such as jazz, pop, folk, and blues recorded before the 1960s is arguably limited by the recording techniques of the time as well as the degradation of physical media. Furthermore, modern recordings can also suffer from diminished audio quality, due to the use of lossy compression, downsampling, packet loss, or clipping. In the broadest sense, audio enhancement aims to restore a degraded signal to improve its sound quality (Godsill et al., 2002). As such, audio enhancement may target the removal of noise, the suppression of cracks or pops (e.g. from an old vinyl record), signal completion to fill in gaps (so-called "audio inpainting" (Adler et al., 2012; Perraudin et al., 2018)), or the extension of the bandwidth from a band-limited signal.

To transmit audio signals through internet streams, or for the ease of storing, common operations such as compression, bandwidth reduction, and low-pass filtering all result in the removal of at least part of the high-frequency audio content. While this process can be understood as a relatively straightforward mapping from a *full-bandwidth*, or *wideband* signal to a *band-limited* or *narrowband* signal, the corresponding inverse problem, namely *bandwidth extension*, seeks to reconstructing missing high-frequency content and is thus non-trivial. Nevertheless, bandwidth extension is crucial for increasing the fidelity of audio, especially for speech and music signals.

We move on to discuss the datasets and the methods employed in audio bandwidth extension.

2.7.1 Datasets

Audio bandwidth extension models take a band-limited audio input signal and create their full-band counterparts. Band-limited audio refers to audio signals whose frequency content is restricted to a certain range, typically lower than the full audible spectrum (20 Hz - 20 kHz). Since signals are converted from analog to digital using sampling, the highest representable frequency is directly related to the digital signal's sampling rate. The Nyquist theorem statest that the minimum sampling rate required to accurately reconstruct an analog signal is twice the highest frequency component present. Therefore, in audio processing, full-band signals are commonly sampled at 44.1 or 48 kHz.

To study the task of audio bandwidth extension, existing models typically use full-band (44.1+ kHz) audio as ground truth, with their band-limited, i.e., low-pass filtered counterparts serving as input. Several music source separation datasets include full-band audio and are well-suited for this task. Although these datasets provide isolated stems for individual instruments, the availability of full-band mixtures makes them particularly useful for audio bandwidth extension research. The DSD100 (Demixing Secrets Dataset) dataset contains 100 music tracks spanning various styles, along with isolated stems for drums, bass, vocals, and other sources (Liutkus et al., 2017). Similarly, the MUSDB18 dataset includes 150 full-band tracks from diverse genres, each with stems for drums, bass, vocals, and other instruments (Rafii et al., 2019). The MedleyDB dataset offers 254 full-band songs and their individual tracks, although the stems are categorized with finer and more variable labels, such as audience noise from live performances or individual drum components like kick drum (Bittner et al., 2016).

The Open Dataset of Audio Quality (ODAQ) features 240 audio samples that are sampled at 44.1 or 48 kHz (Torcoli et al., 2024). It features streaming audio provided by Netflix[10] and Fraunhofer IIS[11]. This dataset also includes quality-degraded versions of each sample, aimed at audio enhancement research. The ICASSP (International Conference on Acoustics, Speech, and Signal Processing) 2023 Acoustic Echo Cancellation (AEC) Challenge introduces a full-band speech dataset with 10,000 samples and versions with synthetic echo effect added (Cutler et al., 2023). The Synthetic Polyphonic Ambient Sound Source (SPASS) Dataset presents 25,000 audio samples with a sampling rate of 44.1 kHz. This dataset mixes short audio events with longer environmental sounds, and is aimed at audio event detection (Viveros-Muñoz et al., 2023).

Table 2.4 summarizes the full-band audio datasets, their intended tasks, and sample sizes.

2.7.2 Methods

The first applications of audio bandwidth extension addressed speech signals only, due to the practical problems arising from the low bandwidth of the telephone systems. One of the earliest works uses a statistical approach in which narrowband and wideband spectral envelopes which are assumed to be generated by a mixture of narrowband and wideband sources (Cheng et al.,

[10] https://netflix.com
[11] https://iis.fraunhofer.de

Table 2.4: Full-band audio datasets

Dataset	Task	Sample size
MedleyDB (Bittner et al., 2016)	Music source separation	254
DSD100 (Liutkus et al., 2017)	Music source separation	100
MUSDB18 (Rafii et al., 2017)	Music source separation	150
SPASS (Viveros-Muñoz et al., 2023)	Audio event detection	25,000
AEC (Cutler et al., 2023)	Acoustic echo cancellation	10,000
ODAQ (Torcoli et al., 2024)	Streaming audio enhancement	240

1994). The probability of each source contributing to the speech signal is parameterized and then estimated using the expectation maximization (EM) algorithm. Mapping-based techniques aim at learning the associations between features belonging to the narrowband and wideband speech where features including linear predictive coding (LPC) coefficients and line spectral frequencies (LSFs). Codebook mapping-based methods make use of two learned codebooks, belonging to the narrowband and wideband signals, containing spectral envelope features, where a one-to-one mapping exists between their entries (Epps & Holmes, 1999; Yoshida & Abe, 1994). For each frame of the input signal, the best matching entry in the narrowband codebook is found and then mapped to the corresponding entry in the wideband codebook. A similar approach is used in linear mapping-based methods, where narrowband features are mapped to wideband features using a linear transformation, where the transformation matrix is learned using methods such as least-squares (Chennoukh et al., 2001; Nakatoh et al., 1997).

Later methods sought to learn to model the wideband signal directly, rather than the mapping between predefined features. Gaussian mixture models (GMMs) have been used to estimate the joint probability density of narrowband and wideband signals (Nour-Eldin & Kabal, 2008; Park & Kim, 2000b). The parameters of the model, namely prior probabilities, mean vectors, and covariance matrices are learned from a training dataset. More sophisticated approaches include the use of hidden Markov models (HMMs), where each state of the model represents the wideband extension of its narrowband input (Bauer & Fingscheidt, 2008; Jax & Vary, 2003; Song & Martynovich, 2009). Due to its recursive mechanism, HMMs can leverage information from the past input frames. Methods based on non-negative matrix factorization (NMF) model the speech signals as a combination of learned non-negative bases (Bansal et al., 2005; Sun & Mazumder, 2013). In the testing stage, low-frequency base components of the input can be used to estimate how to combine the high-frequency base components to create the wideband signal. Finally, the first works using neural networks for speech bandwidth extension employ multilayer perceptrons (MLPs) to estimate LPC coefficients of the wideband speech signal (Iser & Schmidt, 2003), or to find a shaping function which transforms the spectral magnitude (Kontio et al., 2007). We note that these early works used very small neural networks, in which the total number of parameters was around 100.

More recent approaches to audio bandwidth extension use DNNs, with many more layers and far greater representation power than their older counterparts. DNNs also eliminate the need for

hand-crafted features, as they can use raw audio or time-frequency transforms as input, and then learn appropriate intermediate representations. One of the earliest works using DNNs for audio bandwidth extension works with the frequency spectrogram (K. Li et al., 2015). A much deeper model employs the popular *U-Net* architecture (Ronneberger et al., 2015) and works in the raw audio domain, performing experiments on both speech and single instrument music (Kuleshov et al., 2017). T.-Y. Lim et al. (2018) combines the two aforementioned approaches, resulting in a dual network, which operates separately in the time and frequency domains, and creates the final output using a fusion layer. To increase the qualitative performance, namely, the clarity of the produced audio, generative adversarial networks (Goodfellow et al., 2020) are also employed in DNN-based audio bandwidth extension (S. Kim & Sathe, 2019; Sautter et al., 2019).

While the enhancement of old music recordings can be partially framed in the context of bandwidth extension, certain risks arise when considering the data that DNNs are given for training. Even though trained DNNs can perform well on samples from the training data, they may not exhibit the same performance on unseen samples from testing data. This phenomenon is named *sample overfitting* and even though it is an important concern, especially for classification tasks, its existence in generative tasks, such as image super-resolution, audio bandwidth extension, and adversarial generation, is debated. Recent studies show that sample overfitting is not observed for both discriminators and generators of generative adversarial networks (Adlam et al., 2019; Q. Xu et al., 2018), and supervised generative networks for video frame generation (Sulun, 2018; Sulun & Tekalp, 2021). Furthermore, state-of-the-art image super-resolution networks do not include any regularization layers, such as batch normalization (Ioffe & Szegedy, 2015) and dropout (Srivastava et al., 2014), to avoid overfitting (Fan et al., 2017; B. Lim et al., 2017; Y. Zhang et al., 2018).

2.8 Overview of the following chapters

This concludes our discussion of the background and transitions us into presenting our novel contributions and overall methodology. We begin by developing a standalone conditional music generator, independent of video input. Initially, the model is conditioned on emotional and temporal signals. However, emotion-based music generation requires symbolic music data annotated with emotion labels. As discussed in Section 2.4.3.1, existing emotion-labeled MIDI datasets are too small to effectively train deep neural networks. Therefore, in Chapter 3, we address this limitation by creating a large-scale emotion-labeled MIDI dataset.

With this dataset in place, we focus on conditional music generation in Chapter 4. After establishing a standalone music generator, we shift our attention to video analysis in Chapter 5. In Chapter 6, we describe how we integrate the conditional music generator with the video analysis module. We conclude our methodological discussion in Chapter 7, where we explore the challenges of audio generation, specifically audio bandwidth extension. Finally, in Chapter 8, we present our conclusions and outline directions for future research.

Chapter 3

Emotion labeling of MIDI

Conditional deep neural networks are trained to generate outputs based on specific input conditions. Training such models requires a dataset that includes both the target outputs and their corresponding conditions. During training, the condition is provided as input to the DNN, which then generates an output. The model is optimized by minimizing a loss function that measures the difference between the generated output and the ground-truth target. During inference, only the condition is needed, as the trained DNN can produce an output that corresponds to the given condition.

A core component of our video-based music generator is the conditional music generator. Although the conditioning inputs are ultimately derived from video in later stages of our work, we begin by designing and testing the music generator with manually provided conditions.

Our ultimate goal is to establish a correspondence between input video and output MIDI by leveraging both high-level emotional features and low-level temporal features. Low-level temporal features can be directly extracted from the MIDI data itself, so this stage does not require any additional labels. In contrast, MIDI files do not contain high-level emotion information. Therefore, to train an emotion-conditioned MIDI generator, we require MIDI datasets annotated with emotion labels. Later in Chapter 4, we describe how these labels are utilized during MIDI generation.

As our goal is to train deep neural networks for improved music generation, we require large datasets. However, as discussed in Section 2.4.3.1, existing emotion-labeled MIDI datasets are too limited in size to support such training. To address this, we provide emotion annotations for large-scale MIDI-only datasets. Given the scale of these datasets, manual labeling is impractical. Instead, we propose an automatic labeling approach that leverages additional modalities such as audio and text. Specifically, we first utilize the alignment between MIDI and audio versions of songs. Then, we explore potential emotional correlations between a song's MIDI representation and its lyrics.

Spotify[1] provides both low- and high-level features for the songs in its catalog. Our analysis indicates that features can be effective for representing musical emotion. Furthermore, since

[1] https://spotify.com

some MIDI files include embedded lyrics, we can also leverage text processing models to extract emotional information from the lyrical content. Next, we detail these two approaches.

3.1 Using Spotify features

The Spotify for Developers API (Application Programming Interface)[2] provides automated access to audio features of songs. Through this API, we retrieve various musical attributes, including danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, and tempo. Among these, *valence* is especially relevant for representing musical emotion. Although the exact methods used to compute these features are not publicly disclosed, detailed descriptions are available in the official documentation[3].

Using this API, our goal is to label the Lakh MIDI Dataset (LMD), the largest available collection of MIDI files. To do so, we must first identify corresponding entries in Spotify's catalog. Fortunately, LMD includes a subset called *LMD-matched*, which aligns a portion of its MIDI files with entries from the Million Song Dataset (MSD) (Bertin-Mahieux et al., 2011). Each MSD entry provides metadata such as song title, artist name, and identifiers that link to another music database, The Echo Nest[4]. The Million Song Dataset Echo Nest mapping archive[5] extends this connection by providing Spotify track IDs for the Echo Nest. In summary, we follow this chain of mappings—starting from LMD-matched, passing through MSD and Echo Nest—to ultimately associate MIDI files with their corresponding Spotify tracks.

However, not all entries in the MSD have an associated Spotify track ID. To expand our labeled dataset, we adopt an alternative strategy: we use the artist names and song titles provided in the MSD to perform direct searches in Spotify's database.

The presence of the *valence* feature within Spotify's audio attributes allows us to directly integrate it into the circumplex model of affect (Section 2.2.2, Figure 2.4) (Russell, 1980), which uses valence and arousal to represent the emotion of a MIDI track. This simplifies our task to modeling the arousal feature. However, our initial analysis of the Spotify features suggests that the *energy* feature is not an ideal fit for modeling arousal. We hypothesize that this discrepancy arises from the timbral differences between audio and symbolic music, meaning that the same composition can be performed with varying energy levels (e.g., volume, intonation, effects) depending on the recording.

To model the arousal dimension, we draw on existing literature and use low-level features that can be directly extracted from MIDI (H. H. Tan & Herremans, 2020; Williams, Kirke, Miranda, et al., 2015). Our experiments show that tempo and note density are particularly effective in representing arousal.

We estimate the tempo using the built-in metadata of the MIDI file. The tick scale indicates the temporal resolution of the MIDI file. For instance, a tick scale of 0.001 means that MIDI events

[2] https://developer.spotify.com
[3] https://developer.spotify.com/documentation/web-api/reference
[4] https://en.wikipedia.org/wiki/The_Echo_Nest
[5] https://labs.acousticbrainz.org/million-song-dataset-echonest-archive

can be separated by at least 1 millisecond. The resolution refers to the number of ticks per musical beat. In music, a beat serves as the basic unit of time, often represented as a rhythmic pulse that defines the tempo and structure of a piece. By multiplying the tick scale by the resolution, we obtain the duration of a beat in seconds. To estimate the tempo in beats per minute (BPM), we divide the duration of one minute by the beat duration. The following equation summarizes this approach:

$$\text{Estimated tempo} = \frac{60.0}{\text{Tick scale} \times \text{Resolution}} \tag{3.1}$$

We manually reviewed a set of samples to estimate their tempo by ear and compared these estimates with our computed results. We found that the estimation aligns well when the MIDI file includes a drum track, which is arguably the most influential instrument for establishing rhythm. However, our method occasionally fails in the absence of a drum track, particularly in piano-only pieces. Later in our emotion-based music generation research we experiment with using estimated tempo as arousal only when a drum track is present.

We also extract the note density, which is defined as the average number of notes played by all instruments per second. Additionally, we calculate the number of instruments, and the average note density, which is obtained by dividing the note density by the number of instruments. Using average note density as the arousal feature also works even when there are no drum tracks. However, a manual analysis of the outputs of conditional music generation shows that using average density as arousal is less reliable compared to using the estimated tempo.

Our entire dataset creation pipeline is illustrated in Figure 3.1.

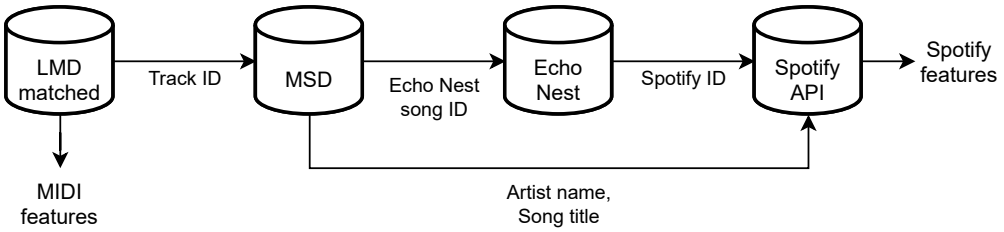


Figure 3.1: Dataset creation pipeline.

For completeness and to foster future research, we derive the low-level MIDI features for the entire Lakh MIDI dataset, labeled as *LMD-full*, where not all samples are necessarily mapped to the entries in the MSD. The LMD-full dataset consists of 178,561 MIDI files. After investigation, we find that 174,270 of these were valid, and we discard the remaining corrupt or empty files. The Lakh MIDI dataset was constructed by downloading MIDI files from publicly available sources on the internet and then retaining unique files based on their hash values. However, upon further examination, we find that MIDI files with different hash values can still have the same musical content, possibly due to differences in metadata. To further filter the data and retain only MIDI files with unique musical content, we first convert the MIDI files to pianorolls using

the pretty_midi package[6]. We then concatenate the 2-dimensional piano roll matrices of all instruments alphabetically to create a single 2-dimensional matrix that describes only the musical content. After recalculating the hash values using these matrices, we select the files with unique hashes. This process resulted in 152,968 MIDI files with unique musical content. These are the files we ultimately use to train our music generator model.

The matched split of the Lakh MIDI dataset, LMD-matched, consists of 31,034 tracks from the MSD matched with 116,189 MIDI files from the LMD. Since multiple MIDI files can be matched to the same track, and multiple tracks to the same MIDI file, we applied filtering to retain only MIDI files with unique musical content, just as we did for the LMD-full split. Additionally, we kept only the best-matching track from the MSD for each MIDI file, based on the matching scores provided in the LMD-matched subset, ensuring that each MIDI file has a single set of labels. After filtering for valid MIDI files with unique content, we obtained 36,545 MIDI files matched with MSD entries. Using the metadata from the MSD, we queried Spotify's dataset and successfully retrieved audio features for 34,791 of these MIDI files.

In its complete form, our dataset contains high-level labels such as danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, and valence; as well as low-level MIDI features such as note density, tempo and the number of instruments. A comparison between our dataset and existing MIDI datasets with emotion labels is shown in Table 3.1. This dataset forms our training dataset for emotion-based MIDI generator, which is discussed in Section 4.2. We show a sample entry and the included features in Listing 3.1.

Table 3.1: Emotion-labeled MIDI datasets

Dataset	Number of songs	Duration (hours)	Instrumentation	Emotion label
MIREX-like (Panda et al., 2013)	193	11.6	Multi-instrument	Categorical
VGMIDI (Ferreira & Whitehead, 2019)	204	6.4	Piano-only	Dimensional
EMOPIA (Hung et al., 2021)	387	11.0	Piano-only	Categorical
Ours	34791	2161.9	Multi-instrument	Dimensional

[6]https://craffel.github.io/pretty-midi

Listing 3.1: A sample entry from proposed dataset.


```
"cc992d0d8e82d09b7fe2466cf851497a": {

"midi_features": {
"note_density": 30.364985431879415,
"tempo": 84.000084000084,
"n_instruments": 10
},
"matched_features": {
"track_id": "TRUHHPK12903CBA84F",
"match_score": 0.7362919446232072,
"song_id": "SOSYWZT12AB0187E45",
"title": "In The Summertime",
"artist": "Mungo Jerry",
"release": "Uber 30 - das rockt!",
"spotify_id": "5VPOrzHyuULaiCKnwQNNCN",
"spotify_title": "In The Summertime",
"spotify_artist": "Mungo Jerry",
"spotify_album": "Uber 30 - das rockt!",
"spotify_audio_features": {
"danceability": 0.681,
"energy": 0.509,
"key": 4,
"loudness": -8.504,
"mode": 1,
"speechiness": 0.0461,
"acousticness": 0.497,
"instrumentalness": 2.72e-06,
"liveness": 0.188,
"valence": 0.963,
"tempo": 82.614,
"type": "audio_features",
"id": "5VPOrzHyuULaiCKnwQNNCN",
"uri":
"spotify:track:5VPOrzHyuULaiCKnwQNNCN",
"track_href": "https://api.spotify.com/v1/
                tracks/5VPOrzHyuULaiCKnwQNNCN",
"analysis_url":"https://api.spotify.com/v1/
                 audio-analysis/5VPOrzHyuULaiCKnwQNNCN",
"duration_ms": 210387,
"time_signature": 4
}
}
}
```

3.2 Emotion classification of song lyrics

Our second method for obtaining emotion labels for MIDI samples involves analyzing the emotions in the song lyrics included in the MIDI files. While MIDI files are primarily designed to contain note events, they can optionally include text-based metadata such as track name, copyright notice, and lyrics.

Our approach takes advantage of the natural connection between lyrics and music, both of which are often tied to emotions (Stratton & Zalanowski, 1994). Similar to the Spotify music features, we note that the emotional content of lyrics can also provide weak features for the MIDI samples, though there are instances where the lyrics and music convey conflicting emotions. A notable example is the song *Every Breath You Take* by The Police, where the music conveys a loving and caring tone, while the lyrics are written from the perspective of a stalker, reflecting unhealthy fixation, jealousy, and control[7].

While the Spotify Developers API directly provides valence values, emotional features from lyrics must be extracted through text analysis. Our approach is to use a text emotion classifier to analyze the song lyrics and assign the predicted emotions to the MIDI samples. To this end, we first train models for emotion classification on the GoEmotions dataset, one of the largest text datasets with 28 fine-grained emotion labels (Demszky et al., 2020).

3.2.1 Training

The first step toward our aim of building an emotion-labeled symbolic music dataset is training a model to perform multi-label emotion classification based on text input.

3.2.1.1 Dataset

We train our model using the GoEmotions dataset (Demszky et al., 2020). This dataset consists of English comments from Reddit[8], which are manually annotated to identify the underlying emotions. It is a multi-label dataset, meaning each sample can have more than one emotion label. The dataset includes 27 emotions and a "neutral" label. Demszky et al. (2020) further grouped the labels into 7 categories, including the six basic emotions identified by Ekman (1971) (joy, anger, fear, sadness, disgust, and surprise) as well as the "neutral" label, as shown in Table 3.2 . The dataset contains a total of 58k samples, which are split into training, validation, and testing sets in a ratio of 80%, 10%, and 10%, respectively. Given the number of labels and its size, GoEmotions is one of the largest emotion classification datasets and has the highest number of discrete emotion labels (Kusal et al., 2022).

[7] https://www.bbc.co.uk/radio2/soldonsong/songlibrary/indepth/everybreathyoutake.shtml
[8] https://www.reddit.com

Table 3.2: Mapping of GoEmotions labels to Ekman categories

GoEmotions labels	Ekman categories
anger, annoyance, disapproval	anger
disgust	disgust
fear, nervousness	fear
admiration, amusement, approval, caring, desire, excitement, gratitude, joy, love, optimism, pride, relief	joy
sadness, disappointment, embarrassment, grief, remorse	sadness
confusion, curiosity, realization, surprise	surprise
neutral	neutral

3.2.1.2 Implementation details

We employ DistilBERT as the backbone of our model (Sanh et al., 2019), which is a condensed and compressed variant of the BERT (Bidirectional Encoder Representations from Transformers) model (Devlin et al., 2019), achieved through knowledge distillation (Bucila et al., 2006; Hinton et al., 2015). DistilBERT utilizes fewer layers than BERT and learns from BERT's outputs to mimic its behavior. Our model consists of 6 layers, with each layer containing 12 attention heads and a dimensionality of 768, yielding a total of 67M parameters. To facilitate multi-label classification, we customize the output layer while adding a sigmoid activation layer at the end, as opposed to using a softmax layer for single label classification. The output layer's size is determined by the number of labels present in the training dataset, which can be either 7 or 28.

We train our models using binary cross-entropy loss. For evaluation, we use precision, recall, and F1-score, with macro averaging. We search for the optimal decision cutoff to maximize the F1-score. The decision cutoff is set at 0.3, meaning that predictions with a value of 0.3 or greater are considered positive predictions and others negative.

We train two models to classify a given text into 7 and 28 labels. We use a dropout rate of 0.1 and a gradient clipping norm of 1. The batch size was set to 16 for the model with 7 output labels and to 32 for the model with 28 output labels. We apply a learning rate of 5e-5 for the former and 3e-5 for the latter. We use early stopping considering the F1-score on the validation dataset, which corresponded to training for 10 epochs for both models. We implement the models using Huggingface library (Wolf et al., 2019) with Pytorch backend (Paszke et al., 2019) and train them using a single Nvidia GeForce GTX 1080 Ti GPU.

3.2.2 Inference

After training the models for text-based emotion classification, we use it in inference mode, using the song lyrics from the MIDI files as inputs. This allowed us to create a MIDI dataset labeled with emotions. Similar to using audio-related labels in Section 3.1, we consider these as "weak labels"

because they are derived from text rather than MIDI, and generated by a trained model rather than annotated by humans.

We use two MIDI datasets that are publicly available and were created by gathering MIDI files from various online sources: the Lakh MIDI dataset consisting of 176k samples (Raffel, 2016) and the Reddit MIDI dataset containing 130k samples[9]. We filter the datasets by selecting MIDI files that contain lyrics in the English language with at least 50 words. This filtering process resulted in a total of 12509 files, consisting of 8386 files from the Lakh MIDI dataset and 4123 files from the Reddit MIDI dataset. During inference, we utilized the two pretrained models, feeding the entire song's lyrics, using a truncation length of 512 tokens.

3.2.3 Results

In this section, we first present the emotion classification performance of our trained models. Then, we introduce the emotion-labeled MIDI dataset, which we created by analyzing the sentiment of the song lyrics using our trained models.

3.2.3.1 Emotion classification on the GoEmotions dataset

We evaluate the performance of our trained models on the test split of the GoEmotions dataset and compared our results with the baseline presented in the original paper (Demszky et al., 2020). Similar to the original paper, we report our results for scenarios using two sets of labels, with 7 and 28 emotions. For each label, we report the precision, recall, and F1-scores along with the macro-averages. It is important to mention that, as the dataset is imbalanced, macro-averaging is more appropriate than micro-averaging, as it was also used in the original paper. We note that the baseline model is BERT and has twice the size of our model (Devlin et al., 2019).

The trade-off between precision and recall is determined by the cutoff value. Therefore, we emphasize higher F1-scores because they provide a more balanced perspective by taking the harmonic mean of precision and recall, and are much less sensitive to the cutoff value. Although the original paper does not state the cutoff value, we achieve the best F1-score and similar performance to the original paper on the 7-label dataset using a cutoff value of 0.3. For consistency, we use the same value for the 28-label dataset. We present our results on the dataset with 7 and 28 labels in Tables 3.3 and 3.4, respectively. Higher values indicate better performance, and the best results are highlighted in bold.

Based on the F1-scores, our model performs comparably to the baseline on the 7-label dataset. Specifically, our model has a better performance on 2 labels, worse on 2 labels, and the same on 3 labels, as well as for the macro-average. On the 28-label dataset, our model surpasses the baseline with only a lower performance on 2 labels, equal performance on 4 labels, and better performance on the remaining 22 labels. Furthermore, our model demonstrates an improvement of 0.04 in terms of the macro-average.

[9] https://archive.org/details/themagicofmidiv1

Table 3.3: 7-label classification results

	Precision		Recall		F1-score	
	Baseline	Ours	Baseline	Ours	Baseline	Ours
anger	0.50	0.50	0.65	0.67	**0.57**	**0.57**
disgust	0.52	0.57	0.53	0.49	**0.53**	0.52
fear	0.61	0.57	0.76	0.73	**0.68**	0.64
joy	0.77	0.75	0.88	0.89	**0.82**	**0.82**
neutral	0.66	0.63	0.67	0.75	0.66	**0.68**
sadness	0.56	0.57	0.62	0.67	0.59	**0.61**
surprise	0.53	0.59	0.70	0.62	**0.61**	**0.61**
macro-average	0.59	0.60	0.69	0.69	**0.64**	**0.64**

Table 3.4: 28-label classification results

	Precision		Recall		F1-score	
	Baseline	Ours	Baseline	Ours	Baseline	Ours
admiration	0.53	0.65	0.83	0.75	0.65	**0.70**
amusement	0.70	0.72	0.94	0.91	0.80	**0.81**
anger	0.36	0.53	0.66	0.49	0.47	**0.51**
annoyance	0.24	0.40	0.63	0.31	0.34	**0.35**
approval	0.26	0.39	0.57	0.38	0.36	**0.39**
caring	0.30	0.37	0.56	0.46	0.39	**0.41**
confusion	0.24	0.52	0.76	0.42	0.37	**0.47**
curiosity	0.40	0.47	0.84	0.62	**0.54**	0.53
desire	0.43	0.66	0.59	0.42	0.49	**0.51**
disappointment	0.19	0.39	0.52	0.22	**0.28**	**0.28**
disapproval	0.29	0.39	0.61	0.41	0.39	**0.40**
disgust	0.34	0.64	0.66	0.39	0.45	**0.48**
embarrassment	0.39	0.72	0.49	0.35	0.43	**0.47**
excitement	0.26	0.43	0.52	0.47	0.34	**0.45**
fear	0.46	0.60	0.85	0.76	0.60	**0.67**
gratitude	0.79	0.88	0.95	0.92	0.86	**0.90**
grief	0.00	0.00	0.00	0.00	**0.00**	**0.00**
joy	0.39	0.59	0.73	0.61	0.51	**0.60**
love	0.68	0.78	0.92	0.85	0.78	**0.81**
nervousness	0.28	0.45	0.48	0.43	0.35	**0.44**
neutral	0.56	0.61	0.84	0.76	**0.68**	**0.68**
optimism	0.41	0.56	0.69	0.52	0.51	**0.54**
pride	0.67	0.83	0.25	0.31	0.36	**0.45**
realization	0.16	0.39	0.29	0.14	**0.21**	**0.21**
relief	0.50	0.00	0.09	0.00	**0.15**	0.00
remorse	0.53	0.59	0.88	0.86	0.66	**0.70**
sadness	0.38	0.57	0.71	0.60	0.49	**0.59**
surprise	0.40	0.56	0.66	0.50	0.50	**0.53**
macro-average	0.40	0.53	0.63	0.50	0.46	**0.50**

We hypothesize that a smaller model, such as ours (DistilBERT), may perform better than a larger baseline model (BERT) in certain settings, such as when there are a limited number of training samples or a high output/target dimensionality, as in the case of the 28-label dataset. In these scenarios, models are more prone to overfitting, as has been previously observed (Yu et al., 2017). Additionally, the original paper (Sanh et al., 2019) demonstrates that the DistilBERT model outperforms BERT on the Winograd Natural Language Inference (WNLI) dataset (Levesque et al., 2012).

3.2.3.2 Inference on MIDI datasets

We use our trained models to analyze the song lyrics of the Lakh and Reddit MIDI datasets, resulting in an augmented dataset that contains the file paths to 12509 MIDI files and their corresponding predicted probabilities for emotion labels. To provide more flexibility to the users, we do not apply a threshold to the predicted probabilities and provide the raw classification outputs, allowing the entire dataset to be used as is. We generate two CSV (comma-separated values) files containing the 7 and 28 emotion labels as columns, with the 12509 MIDI file paths as rows.

For demonstration purposes, we provide transposed versions of the tables, using only 3 samples, shown in Tables 3.5 and 3.6. For further demonstration and ease of analysis, we provide excerpts from the lyrics of each of the three sample songs in Listing 3.2, along with the emotions having predicted probabilities higher than 0.1 in descending order. It is noteworthy that having a dataset with 28 emotion labels allows for a more nuanced representation of emotions. For instance, when we examine this dataset, the song "Imagine" is predicted to have “optimism" as its top emotion, whereas "Take a Chance on Me" is predicted to have "caring" as its top emotion. However, both songs are predicted to have "joy" as their top emotion in the dataset with only seven labels.

We also present the number of samples containing each emotion in our datasets in Figure 3.2. In these figures, we exclude the "neutral" label. For the sake of demonstration, we also consider emotions with a prediction value higher than 0.1 as positive labels, meaning that those emotions are present for a given sample. Due to space limitations, the file paths are replaced with the artist and song names and are as the following: John Lennon - Imagine: "`lakh/5/58c076b72d5115486c09a7d9e6df1029.mid`" (artist and title obtained using Million Song Dataset (Bertin-Mahieux et al., 2011)), ABBA - Take a Chance on Me: "`reddit/A/ABBA.TakeachanceonmeK.mid`", Elvis Presley - Are You Lonesome Tonight: "`reddit/P/PRESLEY.AreyoulonesometonightK.mid`"

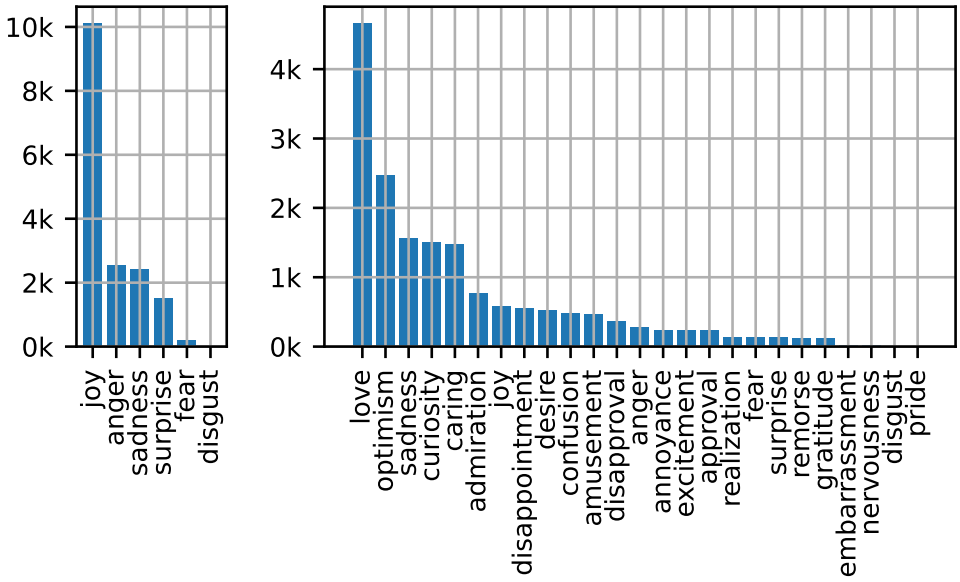


Figure 3.2: The number of samples containing each emotion in our 7-label (left) and 28-label (right) datasets.

Table 3.5: Sample entries from the 7-label dataset.

	John Lennon - Imagine	ABBA - Take a Chance on Me	Elvis Presley - Are You Lonesome Tonight
anger	0.0051	0.0146	0.0272
disgust	0.0003	0.0009	0.0045
fear	0.0005	0.0024	0.0131
joy	**0.8072**	**0.8948**	0.0477
neutral	0.1953	0.1420	0.0782
sadness	0.0013	0.0069	**0.7372**
surprise	0.0754	0.0053	0.5465

Table 3.6: Sample entries from the 28-label dataset.

	John Lennon - Imagine	ABBA - Take a Chance on Me	Elvis Presley - Are You Lonesome Tonight
admiration	0.0021	0.0091	0.0048
amusement	0.0051	0.0012	0.0027
anger	0.0025	0.0018	0.0053
annoyance	0.0024	0.0020	0.0075
approval	0.0026	0.0809	0.0072
caring	0.0067	**0.6169**	0.0601
confusion	0.0070	0.0035	0.1029
curiosity	0.0332	0.0141	**0.6502**
desire	0.0482	0.0472	0.0055
disappointment	0.0044	0.0016	0.0199
disapproval	0.0019	0.0030	0.0048
disgust	0.0007	0.0003	0.0009
embarrassment	0.0006	0.0002	0.0045
excitement	0.0130	0.0049	0.0011
fear	0.0026	0.0026	0.0035
gratitude	0.0007	0.0017	0.0059
grief	0.0008	0.0016	0.0085
joy	0.0025	0.0040	0.0018
love	0.0021	0.1079	0.0193
nervousness	0.0007	0.0017	0.0094
neutral	0.2954	0.4288	0.0757
optimism	**0.7554**	0.1423	0.0060
pride	0.0010	0.0013	0.0006
realization	0.0023	0.0040	0.0045
relief	0.0004	0.0033	0.0011
remorse	0.0005	0.0012	0.1491
sadness	0.0011	0.0027	0.1767
surprise	0.0107	0.0005	0.0020

Listing 3.2: Sample entries with excerpts from lyrics, and emotions with a predicted value higher than 0.1.

```
File path: lakh/5/58c076b72d5115486c09a7d9e6df1029.mid
Artist - Title: John Lennon - Imagine
Lyrics:
Imagine there's no heaven.
It's easy if you try.
No hell below us.
Above us, only sky.
Imagine all the people.
Livin' for today.
7-label predictions:
joy:            0.8072
neutral:        0.1953
28-label predictions:
optimism:       0.7554
neutral:        0.2954

File path: reddit/A/ABBA.Take a chance on me K.mid
Artist - Title: ABBA - Take a Chance on Me
Lyrics:
If you change your mind, I'm the first in line.
Honey, I'm still free.
Take a chance on me.
If you need me, let me know, gonna be around.
If you've got no place to go, if you're feeling down.
7-label predictions:
joy:            0.8948
neutral:        0.1420
28-label predictions:
caring:         0.6169
neutral:        0.4288
optimism:       0.1423
love:           0.1079

File path: reddit/P/PRESLEY.Are you lonesome tonight K.mid
Artist - Title: Elvis Presley - Are You Lonesome Tonight
Lyrics:
Are you lonesome tonight?
Do you miss me tonight?
Are you sorry we drifted apart?
Does your memory stray to a bright summer day,
When I kissed you and called you sweetheart?
7-label predictions:
sadness:        0.7372
surprise:       0.5465
28-label predictions:
curiosity:      0.6502
sadness:        0.1767
remorse:        0.1491
confusion:      0.1029
```

3.3 Discussion

In our exploration of automatic emotion labeling for symbolic music, we investigated two primary approaches: utilizing Spotify's audio-related features and analyzing text-based song lyrics embedded within MIDI files. Our objective was to augment large-scale MIDI datasets with emotional metadata to facilitate the training of conditional music generators. Our preliminary evaluations revealed that emotion labels inferred from Spotify features were more reliable than those derived from lyrics. As a result, we ultimately relied on Spotify-derived labels for emotion-conditioned MIDI generation.

Spotify offers a set of high-level musical features, with valence in particular aligning closely with established emotion models such as the circumplex model of affect. We leveraged the alignment between the Lakh MIDI Dataset and Spotify's catalog through the Million Song Dataset and Echo Nest IDs. This process yielded emotional metadata for tens of thousands of MIDI files, offering a practical foundation for emotion-based MIDI generation.

While we also explored using lyrics to infer emotional content, this approach proved less effective in our setting. Despite training a multi-label emotion classifier using the GoEmotions dataset and a DistilBERT backbone, the labels generated from lyrics lacked consistency and failed to produce perceptually coherent results in downstream generation tasks. This limitation may stem from the ambiguity of lyrics, mismatches between lyrical and musical emotion, and the domain gap between internet text and song lyrics. As such, lyric-based annotations were excluded from the final generation pipeline.

Our quantitative results on text-based emotion classification highlights the efficiency of DistilBERT for emotion classification, especially in cases where the number of output labels is high, as in the 28-emotion label set. Notably, our model outperformed the baseline on the 28-label dataset, demonstrating that smaller models can prevent overfitting and still achieve strong results in settings with limited training data and many classes. The ability to classify text into 28 fine-grained emotion categories enabled us to capture more nuanced emotional variations in songs, as shown by the our demonstrative output samples in Listing 3.2.

Although we exclude the lyric-based labels from our music generation pipeline, emotion classification from lyrics remains a promising research direction. Lyrics can convey the songwriter's emotional intent and provide a semantic layer beyond what audio features reveal. With improved domain-specific models or multi-modal strategies that combine lyrics with audio and symbolic data, more nuanced and accurate emotional labeling could be achieved. Our dataset and models are included to support these avenues, particularly for tasks such as mood-based music retrieval or affect-aware lyric composition.

Together, these labeling approaches lay the groundwork for emotion-conditioned music generation, where symbolic music can be composed or guided based on target emotional characteristics. In the next chapter, we describe how these labels are used to train generative models that respond to emotional conditions.

Chapter 4

Conditional music generation

In this chapter, we present our approach to conditional music generation, which is the most important component of our video-based music generation system, as this module is responsible for producing the final musical output. While the input conditions are manually provided in this chapter, in Chapter 6 we replace them with results obtained from video analysis. As previously discussed, we chose to focus on conditional music generation first, and video analysis later in Chapter 5, to ensure musically meaningful results at every stage of our research.

As explained in Chapter 1, our goal is to achieve both high- and low-level alignment between the input video and the generated music. We use emotions as the high-level and temporal boundaries as the low-level intermediary features. This enables the generated soundtrack to reflect the video's emotional tone while maintaining temporal synchronization.

We first outline our method for temporally-conditioned music generation.

4.1 Music generation based on temporal boundaries

Temporal alignment of the video and music is essential to create a "synchronized" feel (Prendergast, 1992). Our approach seeks to overcome the limitations of the dense temporal conditioning mechanisms used by state-of-the-art methods (Di et al., 2021), as discussed in Section 1.1.1. To address this, we propose utilizing sparse temporal features, specifically temporal boundaries. In this approach, we first define what constitutes a temporal boundary in both video and music.

Since scenes (or equivalently, shots) are the fundamental units of a video, we use scene cuts to mark temporal boundaries (Cutting, 2005; Nitanda et al., 2005). In the musical domain, we adopt chord locations as temporal boundaries, as chords frequently function as structural anchors in compositions (Bharucha & Krumhansl, 1983). By aligning these two modalities, our goal is to train a music generator that places chords near the scene boundaries of the input video, while maintaining overall musical coherence throughout the generated melody.

To enable temporally-conditioned music generation, we first extract chord locations to use as input during training. We label chords in our training dataset—the Lakh Pianoroll Dataset-5 (Dong et al., 2018), a version of the Lakh MIDI Dataset (Raffel, 2016) with instruments merged into five

categories: bass, drums, guitar, piano, and strings. We focus on guitar and piano chords containing at least three simultaneous notes lasting a minimum of two beats. We refer to these as *strong chords*, noting that this terminology differs from any potential uses of the term in harmonic contexts. To label these strong chords, we extend our token vocabulary by adding the `CHORD` token. Each strong chord is marked by inserting a `CHORD` token before its first `ON` (i.e., `NOTE-ON`) token. To allow the model to generate chords not only at video scene cut locations but also independently where musically appropriate, we randomly remove 20% of `CHORD` tokens during training.

Chords are integral to a melody, providing harmonic and rhythmic support to surrounding notes, both preceding and following (Bharucha & Krumhansl, 1983). During inference, forcing a `CHORD` token into the sequence at a specific location may cause the chord to sound off-beat or overly abrupt. This occurs because the model, having no prior knowledge of the upcoming chord, may generate preceding notes that do not align naturally with it. To address this, we propose a method that enables the model to "anticipate" upcoming chords and generate preceding notes and time shifts accordingly. Additionally, we train the model to generate the `CHORD` token itself, ensuring rhythmic consistency in the generated music. To achieve this, we define *boundary offsets* for each input token, representing the remaining time until the next boundary. These offsets are capped at a maximum value and, since our music generator is autoregressive, they are computed based on future chords rather than past ones. Furthermore, we amplify the loss of the `CHORD` token by a factor of 10, as it influences multiple preceding chord notes and plays a critical role in structuring the generated music.

In Figure 4.1, we illustrate the concept of boundary offsets relative to the strong chord. The top part of the figure shows the symbolic music, where a chord containing three or more simultaneous notes and lasting longer than a certain threshold defines a musical boundary. The bottom part of the figure displays the boundary offsets, which represent the temporal distance to the next boundary. It is important to note that these figures are illustrative; the offsets do not perfectly align with the music, except at the defined boundary.

We process boundary offsets using a feed-forward network to generate boundary offset encodings, which are then concatenated with learned positional encodings (defined in Section 2.3) (B.

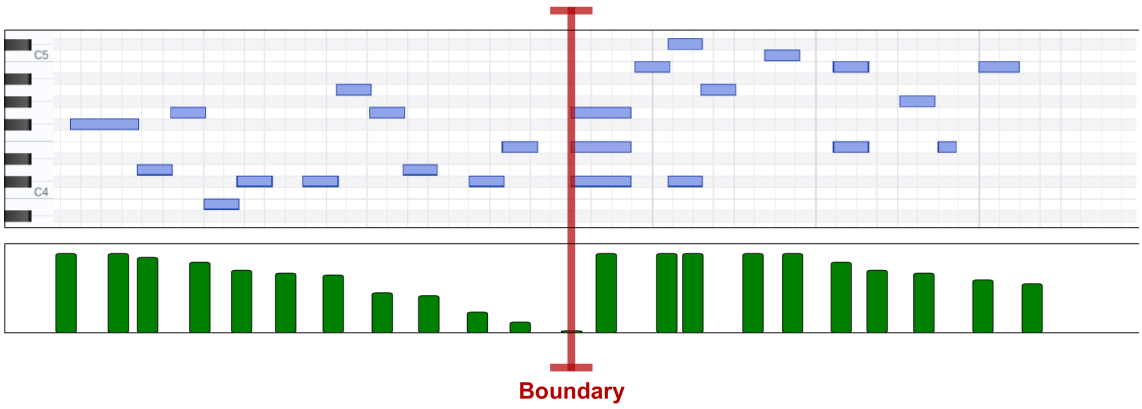


Figure 4.1: Graphical illustration of a musical boundary and its offsets.

Wang et al., 2020) along the feature dimension. This architectural choice is motivated by several key factors. First, we inject boundary offset encodings at the input level rather than within the transformer body, ensuring that the core model remains unchanged. This allows the model to process inputs with or without boundary offsets, enabling seamless fine-tuning by repeating the learned positional encodings along the feature dimension when boundary offsets are absent. Second, we avoid adding boundary offset encodings directly to learned positional encodings to maintain a distinction between the two. Finally, recent studies suggest that decoder-only transformers can implicitly learn positional encodings through their internal weights, even without explicitly adding them (Haviv et al., 2022; Kazemnejad et al., 2024). Based on this insight, we halve the feature length of learned positional encodings and allocate the remaining feature space to boundary offset encodings. The resulting vector sequence consists of positional encodings augmented with boundary offset encodings. Following the standard transformer model, we add this sequence to the token embeddings (Vaswani et al., 2017) before passing it to the transformer body with relative global attention (C.-Z. A. Huang et al., 2019).

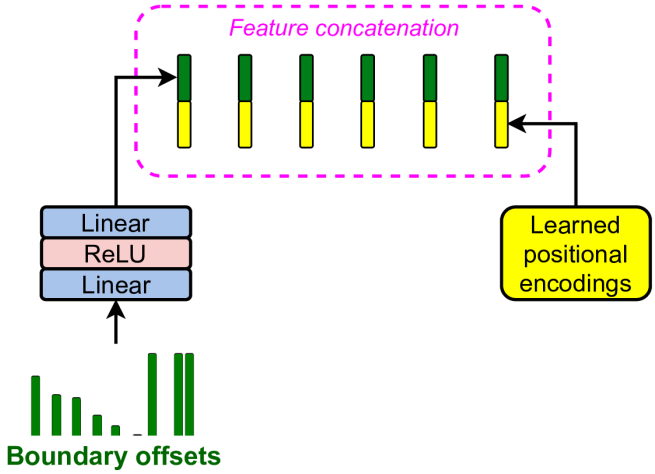


Figure 4.2: Conditioning mechanism based on boundary offsets.

Algorithm 1 outlines the process of computing boundary offsets during inference, i.e., music generation. The model generates music autoregressively, producing one token per forward pass. We maintain a time cursor by tracking the generated `TIMESHIFT` tokens and compute the boundary offset, i.e., the time remaining until the next boundary, for each generated token. If the model generates a `CHORD` token, we calculate the absolute difference between the time cursor and each input boundary. The boundary with a difference smaller than the predefined sensitivity threshold of 1 s is considered successfully generated. We then remove these boundaries from future offset calculations by replacing them with infinity. Next, we compute the boundary offset for the generated token, regardless of its type. This offset represents the distance to the next closest boundary. We compute it by subtracting the time cursor from each input boundary and replacing any negative values (corresponding to past boundaries) with infinity to ignore them. We then take the minimum of the resulting values as the offset to the nearest boundary and clamp it to a maximum value if

Algorithm 1 Creating boundary offsets during music generation in inference.

Input: List of input boundaries (in seconds), **b**; video duration, d; valence, x_v; arousal x_a
Parameter: Sensitivity, ξ; distances to all boundaries, $\boldsymbol{\delta}_b$; maximum offset, δ_{max}; generated boundary mask, $\mathbf{m_b}$; token type t_t; token value, t_v; time cursor, c; generation function including forward-pass and sampling, g
Output: List of generated tokens **t**; list of boundary offsets, $\boldsymbol{\delta}$

Let $c = 0$; $\boldsymbol{\delta} = [\,]$; $\mathbf{t} = [\,]$.
while $c < d$ **do**
  `# generate token as (type, value):`
  $(t_t, t_v) = g(\mathbf{t}, \boldsymbol{\delta}, x_v, x_a)$
  if t_t == `TIMESHIFT` **then**
    $c = c + t_v$
  else if t_t == `CHORD` **then**
    $\mathbf{m_b} = |c - \mathbf{b}| < \xi$.
    $\mathbf{b}\,[\mathbf{m_b}] = +\infty$.
  end if
  $\boldsymbol{\delta}_b = \mathbf{b} - c$
  $\boldsymbol{\delta}_b[\boldsymbol{\delta}_b < 0] = +\infty$
  $\boldsymbol{\delta}$.append(min(min($\boldsymbol{\delta}_b$), δ_{max}))
  t.append((t_t, t_v))
end while

necessary. The resulting boundary offset and generated token are appended to their respective lists to be used in the next timestep. For simplicity, Algorithm 1 is presented for a single sample, but in practice, this operation is performed in minibatches. During training, we preprocess the entire input sequence at once by constructing a time grid instead of a time cursor, allowing us to calculate boundary offsets for all tokens simultaneously. For clarity, the initial list of generated tokens is shown as empty; however, in practice, we begin with a `START` token.

The `ON` tokens that appear after a `CHORD` token and before the next `TIME-SHIFT` token are considered notes of the generated chord. To make the generated chord more distinctive, we increase the velocity of these notes.

While in training, the temporal locations of chords from the ground-truth MIDI serve as input boundaries. As we later detail in Chapter 6, during video-based inference, we replace these boundaries with video scene cut locations.

We continue our discussion with conditioning music generation on emotions.

4.2 Music generation based on emotions

Emotion-based music generation is the second module of our conditional music generation model. Compared to temporal conditioning, emotion-based conditioning aims to establish a higher-level connection between the video and music, focusing on the emotional tone.

To enable emotion-based music generation, we previously obtained emotion labels for MIDI datasets, as described in Chapter 3. These labels serve as conditional inputs during model development. Next, we describe the methods we developed for emotion-based music generation.

The backbone of our models is the music transformer (C.-Z. A. Huang et al., 2019), which is a decoder-only transformer using relative position embeddings. We first design a *vanilla* model that does not use any conditioning. It takes musical tokens, projects them into vector space through an embedding layer, and feeds the output into the music transformer.

We explore several methods for conditioning the music generation process on emotion features, which we refer to as *discrete-token*, *continuous-token*, and *continuous-concatenated*. The *discrete-token* approach represents the state-of-the-art in conditional sequence generation (Hung et al., 2021; Keskar et al., 2019; Payne, 2019), where a discrete control token is prepended to the original sequence—in our case, a tokenized MIDI sequence. However, since our emotion conditions are continuous rather than discrete, we design alternative methods to incorporate these continuous conditions, enabling finer-grained control over the generated music.

In the *discrete-token* approach, we convert the continuous emotion conditions into discrete values by placing them into bins. Specifically, we quantize the condition values using 5 equal-sized bins, with the central bin indexed as 0. The number of bins is chosen to reflect typical verbal quantifiers, such as *very low*, *low*, *moderate*, *high*, and *very high*. The control tokens for valence and arousal are prepended to the music tokens, i.e., they are concatenated in the sequence dimension, before being fed into the transformer model. A key limitation of this method is the potential loss of information due to the binning of continuous values.

In the next approach, called *continuous-token*, we utilize the condition values in their original continuous form. The valence and arousal values are passed through separate linear layers to create condition vectors, which match the feature length of the music token embeddings. These condition vectors are then concatenated with the music token embeddings in the sequence dimension and fed into the transformer. Unlike the *discrete-token* approach, this method avoids the information loss associated with quantization by preserving the continuous nature of the emotion conditions.

Our final approach, *continuous-concatenated*, combines the two continuous condition values into a single vector, which is then repeated across the sequence dimension and concatenated with each music token embedding. This design addresses a limitation of the *continuous-token* method, where condition vectors are treated with equal importance as individual music tokens. We argue that emotional conditions influence the musical sequence globally, unlike individual notes that contribute locally. By embedding the condition vector into every token, we ensure that the emotional context is present throughout the entire sequence, reinforcing its influence at every step. The representations of the models can be seen in Figure 4.3.

For a given sample, if a condition label is not present, i.e. is NaN (Not a Number), we are not able to quantize it into a bin, nor project it into a vector space. In this case, we employ learned embedding instead of projections. As any discrete token is projected into vector space using an embedding layer, this approach is equivalent to having special discrete tokens such as `NO_VALENCE` and `NO_AROUSAL` in model's vocabulary. This approach not only accommodates missing data

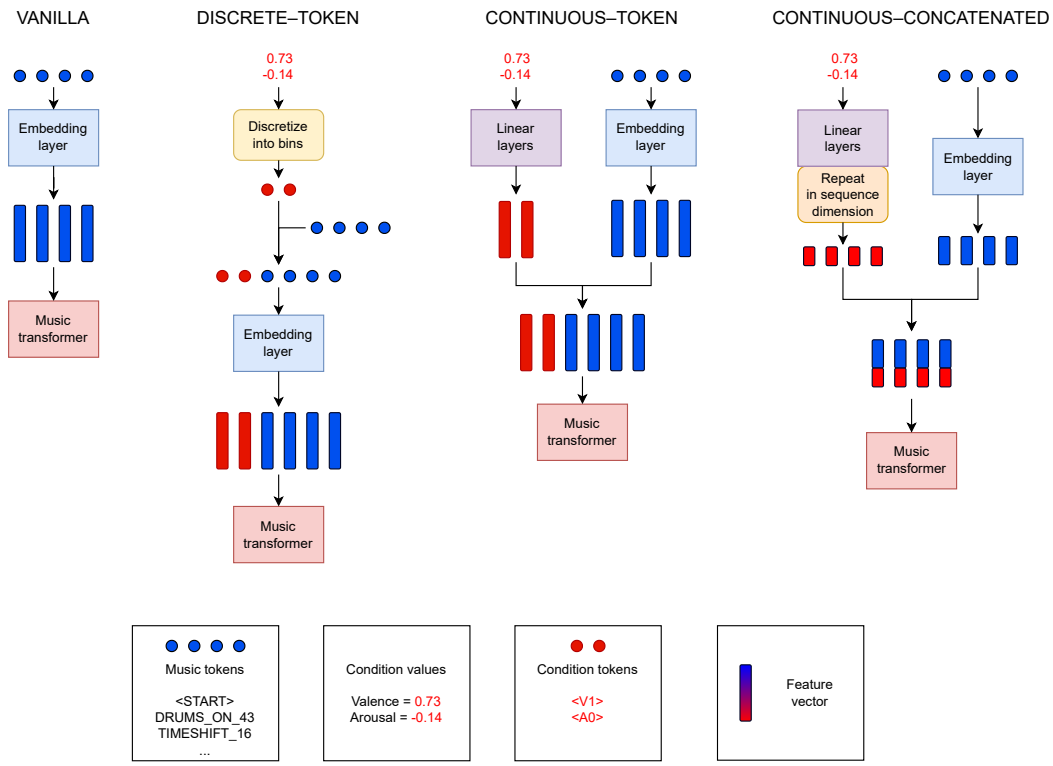


Figure 4.3: Emotion-based music generation models.

but also enables the model to switch between conditional and non-conditional generation within a unified framework.

4.3 Dataset and preprocessing

We train our music generator on the Lakh Pianoroll Dataset (LPD) (Dong et al., 2018), which contains 174,154 pianorolls derived from the Lakh MIDI Dataset (Raffel, 2016). We tokenize the pianorolls using an event-based symbolic music representation (Oore et al., 2020). Specifically, an `ON` (note on) token marks the start of a note, and an `OFF` (note off) token marks its end. These tokens also encode pitch and instrument information. For example, a piano note with a MIDI pitch of 60 (C4) is denoted as `PIANO_ON_60`. We filter out MIDI note-on and note-off events that have a pitch outside the range of the piano, i.e., lower than 21 (A0) and higher than 108 (C8), since these notes aren't audible using standard MIDI soundfonts. Since a larger number of instruments increases vocabulary size, we use the Lakh Pianoroll Dataset-5 variant, where all instrument tracks are merged into five predefined categories: bass, drums, guitar, piano, and strings (Dong et al., 2018). However, our method is adaptable to datasets with different instrument groupings.

We use a temporal resolution of 8 milliseconds with a maximum shift of 1000 milliseconds, as done by Oore et al. (2020). Longer durations are represented using multiple consecutive `TIME-SHIFT` tokens. We also use the `START` tokens to mark the beginning of the songs, the `BAR` tokens to indicate the musical bars, and the `PAD` tokens to standardize input sequence lengths

in minibatches. As mentioned earlier, strong chords are marked by inserting a `CHORD` token before their first `ON` token.

Since we aim to generate multi-instrument compositions, we prioritize pieces with three or more instruments. However, the dataset contains many songs with only one or two instruments. Rather than filtering them out, we prepend special tokens: `FEWER_INSTRUMENTS` for songs with two or fewer instruments and `MORE_INSTRUMENTS` for those with three or more. These tokens allow users to specify instrumentation preferences at inference time while leveraging the entire dataset during training.

As emotion labels, we use our previously constructed Lakh-Spotify dataset, described in Section 3.1. Emotions are represented using the dimensional circumplex model of affect (Russell, 1980), with a slight modification: both valence and arousal are constrained to the range $[-1, 1]$, forming a square valence-arousal space rather than a circular one. Since the original valence values in the Lakh-Spotify dataset span $[0, 1]$, a shift and scale transformation is required. However, we observed an unexpected peak at the 0.0 value in the original valence distribution. Upon examining the corresponding musical samples, this value did not consistently reflect low-valence characteristics, suggesting that missing or undefined values might have been marked as zero. We therefore replace these values with NaN (Not A Number) values, resulting in 23,653 samples with valid valence labels.

To model arousal, we use low-level MIDI features such as note density and tempo, following the approach suggested by Williams, Kirke, Miranda, et al. (2015), and our qualitative viewing of the output showed musically coherent results similar to H. H. Tan and Herremans (2020). We initially experimented with average note density and observed moderate success; however, we ultimately adopted estimated tempo exclusively for samples containing a drum track, as this produced more consistent qualitative results, as explained in Section 3.1. To ensure a robust generation process, we exclude extreme tempo values by setting empirical bounds of 50 and 150, determined through trial and error and manual inspection. Samples falling outside these limits or lacking a drum track are assigned NaN values for arousal. This results in 103,735 samples with valid arousal values. Both valence and arousal values are shifted and scaled to lie within the range $[-1, 1]$. Finally, to create the testing data split, we order the file names from the LPD-matched subset alphabetically, and reserve the last 5%.

4.4 Implementation details

The training input sequences are fixed-sized chunks extracted from the MIDI sequences. In our experiments, we use an input length of 1216 tokens to maximize memory usage. With a probability of 0.05, the input chunk is selected from the beginning of the song, which helps the model learn how songs typically start and better understand the `START` token. With the remaining 0.95 probability, the chunk is taken from a random location in the song, which increases the variability of the inputs.

For data augmentation, we transpose the pitches of all instruments, except for drums, by a randomly chosen integer value between -3 and 3 inclusive. This relatively narrow range ensures that the emotional character of the songs remains intact. By applying these two methods, i.e., random starting points and pitch transposition, the effective training data size becomes significantly larger than the number of songs in the training data split.

Our models have 20 layers and a feature dimension of 768. Each layer has 16 heads and a feed-forward layer with a dimension of 3072. In the *continuous-concat* model, the dimensionalities of the conditioning vectors and token embeddings are 192 and 576, respectively, ensuring that the total dimensionality of the transformer input remains constant at 768 across models. Overall, our models have around 145 million parameters.

We implement our models using the Pytorch library (Paszke et al., 2019) and train them on a single NVIDIA Quadro RTX 6000 GPU. We use the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 2e-5. Our preliminary experiments confirm the findings of Donahue et al. (2019), that common learning rates for language modeling tasks, about 2e-4, are too high for MIDI generation tasks. We reduced the learning rate to 2e-6 when the training loss plateaued and kept training until convergence. We use gradient clipping at a norm of 1, with a dropout rate of 0.1, a batch size of 4, and an input length of 1216 tokens. We use a triangular autoregressive mask to prevent the model from attending to future tokens.

At inference, and before the generation starts, the input sequence only consists of the `[START, BAR]` tokens. Alternatively, to generate a piece of music with a more abrupt beginning, we can omit the `START` token and use only the `BAR` token as the priming sequence. This approach typically results in an output that resembles the middle portion of a song. Additionally, users can choose any MIDI sequence as a primer and have the model continue the given melody.

We generate the output autoregressively, where the generated token is appended to the input sequence, forming the input sequence for the next timestep. When the generated sequence starts to exceed the maximum length of 1216 tokens, we use only the last 1216 tokens of the generated sequence as input, ensuring the length limit is not exceeded. For token generation, we use nucleus sampling with a probability of $p = 0.7$ from a temperature-adjusted softmax distribution (Holtzman et al., 2020), where the temperature is set to 1.2. To prevent excessive repetition, if the number of tokens in the nucleus from the previous step is fewer than 3, we slightly increase the temperature. These hyperparameters are selected through grid search, where different combinations are tested and the results are manually evaluated by listening to the generated music.

We now discuss the evaluation methods for conditional music generation.

4.5 Evaluation

Quantitatively evaluating our temporal conditioning mechanism is challenging. As mentioned earlier, forcing the model to place a chord *exactly* at the conditioning boundary could disrupt the rhythm of the generated music. Instead, our goal is to guide the model to place a chord in the *vicinity* of the temporal boundary, ensuring it fits rhythmically within the rest of the generated

sequence. Moreover, we also want to allow the model to ignore a particular conditioning boundary if it is impossible to place a chord near it without disrupting the rhythm. Therefore, these temporal conditions are soft rather than hard constraints. We do not aim to maximize an accuracy metric; instead, our goal is to achieve a subjective sense of synchronization in the context of video-based music generation. To assess this, we opt for a subjective evaluation where we present the model's outputs overlaid with various input videos. This evaluation is further detailed in Section 6.3, where we use a user survey to gather feedback. One of the survey questions specifically addresses temporal alignment, asking, "How well does the music align with the video in terms of rhythm and timing?" This question helps evaluate the effectiveness of our temporal boundary conditioning method.

We evaluate our emotion-conditioning methods using both qualitative and quantitative approaches. As later detailed in Section 6.3, our video-based user study includes the question: "How well does the music match the video in terms of emotion?" For the quantitative evaluation, we isolate the emotion-conditioning mechanism by disabling temporal conditioning. We evaluate our emotion-conditioning methods using the metrics negative log-likelihood (NLL), top-1, and top-5 accuracies. While measuring top-n accuracy, for each token, the model's output is considered accurate if the ground-truth class is within the top n probabilities of the model's output. The evaluation configuration is the same as the training, namely using chunks with a length of 1216 tokens, and calculating the loss for every single token in the target sequence. This is much more challenging than only predicting the next token given the full sequence, since, at the extreme, the model tries to predict the first note of a song, only given the `START` and `BAR` tokens. We ensure that the entirety of the test split is used by sequentially taking non-overlapping chunks, resulting in 1836 chunks overall.

We additionally perform a quantitative evaluation on samples generated by our conditional models, by analyzing their emotional content, as done by Hung et al. (2021). To this end, we first train a regression model to predict the emotion values of the samples from the training data split. The architecture of the regression model is a music transformer with 8 layers, and the final layer outputs two continuous values, namely the valence and the arousal. Then using the trained conditional generation models, we perform inference using a collection of conditions, and later predict the emotional content of the generated samples using the trained regression model. As the error metric, we use the normalized L_1-distance between the predictions of the regression model and the conditions that were fed during inference. To make a fair comparison against the *discrete-token* model, the condition values are chosen as the midpoints of the bins used by the discrete condition tokens, namely -0.8, -0.4, 0, 0.4, and 0.8. Using a combination of 5 values for valence, and 5 values for arousal, we end up with a collection of 25 condition value pairs. For each model and each condition, we generate 8 samples without "cherry-picking" and report the average error. Each sample has 4096 tokens. The regression model takes inputs with a length of 1216 tokens, similar to the generator. Samples are fed into the regression model using a sliding window with 50% overlap, and outputs are averaged. The overall scheme for evaluating generated samples is visualized in Figure 4.4.

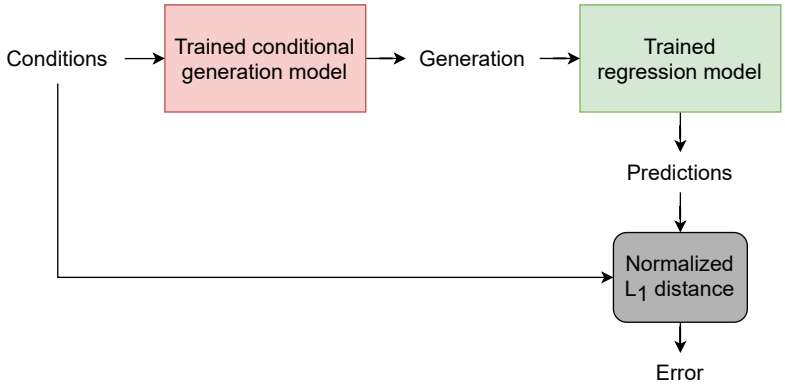


Figure 4.4: Pipeline for evaluation of emotion representation.

We also make the generated samples available online[1]. As explained previously, using the conditional models, we generate 200 samples for each. Since the generated melodies have a fixed number of tokens, their durations in time are inversely proportional to their tempo. The melodies generated with the lowest arousal conditions results in the slowest tempo, and are on average 159.6 seconds long. We also generate 200 more samples using the vanilla model, with no conditioning. Overall, we present 800 samples. The midi files are rendered into mp3 format using the Fluidsynth software and FluidR3_GM soundfont[2].

4.6 Results and discussion

We now present the results and discussion of our emotion-conditioned music generation methods. As explained earlier, the results for the temporal-conditioning mechanism are discussed later in Section 6.4.

The performance of the emotion-conditioning methods according to the prediction accuracy-based evaluation can be seen in Table 4.1. NLL refers to negative log-likelihood, where lower is better. Top-1 and Top-5 refer to the accuracy, where higher is better. The *continuous-concatenated* method outperforms other method, including the state-of-the-art *discrete-token* method, across all metrics.

Table 4.1: Performance of our methods during evaluation.

Model	NLL	Top-1	Top-5
vanilla	0.7445	0.7784	0.9513
discrete-token	0.7375	0.7885	0.9536
continuous-token	0.7122	0.7895	0.9545
continuous-concatenated	**0.7075**	**0.7913**	**0.9548**

[1] https://www.serkansulun.com/midi
[2] https://archive.org/details/fluidr3-gm-gs

In Table 4.2 we show the results for the regression-based evaluation. Error refers to the normalized L_1 distance between conditions fed during inference and the output of the trained regression which consumes the generated samples. We demonstrate that *continuous-concatenated* method outperforms others in terms of its ability to convey emotion. Note, here the vanilla approach is not included since it is not conditioned on any emotion information.

Table 4.2: Performance of our methods during inference.

Model	Error
discrete-token	0.2164
continuous-token	0.1951
continuous-concatenated	**0.1948**

When comparing the performance of the presented methods, we speculate that the main shortcoming of the *discrete-token* and *continuous-token* methods, in contrast to the *continuous-concatenated* method, is their equal treatment of condition values and sequence tokens. While each token in the sequence is primarily useful for making local predictions (i.e., predicting tokens nearby), the condition values have a global impact, as they directly influence the entire generated sample. Our proposed *continuous-concatenated* method maximizes the utility of condition information by incorporating it into each embedding of the input sequence for the transformer. Both of our proposed methods can also use continuous-valued conditions, offering finer control over the generation process.

For reproducibility and to help future research, we open-source our dataset and the code that we used to prepare it[3]. In the NLP community, training large transformers from scratch is a rare practice that is typically replaced by transfer learning, namely by fine-tuning open-source pre-trained models. However, a similar phenomenon does not exist in the field of symbolic music generation. Thus, we additionally open-source our trained models to allow other researchers to cut down on the time and resources for training, with transfer learning. To the best of our knowledge, ours are the largest open-source symbolic music generation models, that are trained on the largest multi-instrument symbolic music dataset, in the literature.

The conditions used in this chapter were either derived from the MIDI data or manually provided during inference. In Chapter 6, we show how these same conditions are automatically extracted from input videos and then passed into our conditional music generator. Before introducing the full video-based system, however, we first describe how these conditions are obtained from video. We now turn to our video analysis methods.

[3]https://github.com/serkansulun/midi-emotion

Chapter 5

Video analysis

In this chapter, we present our video analysis methods, which produce the inputs for the conditional music generator described earlier. We remind that our overall approach follows a backward design: we first develop the music generation component and then construct the video analysis models to ensure musically meaningful outputs at every stage. Specifically, we analyze videos from two complementary perspectives: low-level temporal boundaries and high-level semantics. For the temporal analysis we detect scene cut boundaries, while the semantic analysis involves emotion and genre classification. In the final stage of our research, these video-derived features will enable the generation of music that aligns strong chord boundaries with scene cuts, while also reflecting the video's emotional tone.

As explained in Section 2.5, scene cut extraction is a solved problem. Therefore, we concentrate our research on the more challenging task of semantic video classification. Our video-based music generator relies on a video emotion classifier; however, as noted in Sections 1.2 and 2.5, the limited size of video emotion datasets constrains model complexity and often necessitates using a fixed and small number of input frames. To address this, we first design an emotion classifier suited for these constraints. We then apply the same architectures to trailer genre classification—a task with significantly more data—allowing us to explore the model's capacity beyond the limitations imposed by emotion datasets.

5.1 Scene cut extraction

In our video-based music generator, we use video boundaries in the form of scene cuts, along with high-level emotions. Since scene cut detection is a well-established problem, we use the FFmpeg software for this task. We identify a scene cut when the average pixel difference between consecutive frames exceeds 27%.

To generate musical chords near the scene boundaries, we aim to avoid an excessive number of close boundaries. Therefore, we apply a difference filter to the extracted scene cuts and remove those occurring less than 4 seconds apart. We determined these parameters through trial and error, as well as manual inspection.

We continue with our methodology for semantic video classification.

5.2 Semantic video classification

Both video emotion classification and trailer genre classification fall under the category of semantic video classification. Since both tasks involve using audiovisual data for predictions, we employ similar architectures for both. The video emotion classifier, being integrated into the final video-based music generator, incorporates the most recent updates and adjustments compared to the trailer genre classifier. We highlight these differences where they occur.

Most existing video classification methods use raw pixels and audio as inputs, but this high-dimensional input space can lead to overfitting when data is limited, as discussed in Sections 1.2 and 2.5. Moreover, videos are inherently multimodal, involving elements like objects, scenes, speech, music, sound events, and on-screen text—all of which can contribute to semantic understanding. However, capturing all of these modalities with a single model is a major challenge.

To address this, we use learned features extracted from a variety of pretrained deep neural networks (DNNs). These features are compact, helping to reduce input dimensionality and mitigate overfitting. Because they are derived from models already trained on large datasets, they contain rich, semantically useful information for video analysis. Importantly, we use these pretrained models in forward mode only—without any fine-tuning—significantly lowering memory and computational requirements. We then train lightweight models on top of these features to make predictions, keeping training efficient while still leveraging high-level representations. We now introduce the pretrained models which we use for feature extraction.

5.2.1 Feature extraction

We use the following pretrained models for feature extraction, taking activations from the layer just before the final classification layer, except in the case of text. This approach provides richer and more informative representations than using final predictions alone. For text, we first extract raw output and then process it through additional text models, again using the penultimate layer activations for feature representation.

We note that some of the pretrained models we use also analyze on-screen music. Although our primary task is music generation, we incorporate music analysis models to enhance video classification performance and remain competitive with state-of-the-art methods, while also enabling our classifiers to be applied to tasks that don't involve music generation. Furthermore, our video emotion classifier does not rely on a model dedicated exclusively to music analysis; instead, it uses a unified, state-of-the-art model for general audio analysis. We do not attempt to separate music from other audio sources, as this remains an open research challenge (Pimpale et al., 2016).

CLIP Contrastive Language-Image Pretraining (CLIP) is a state-of-the-art model for image understanding, pretrained using contrastive learning with a large collection of images and their associated captions from the internet (Radford et al., 2021). The pretrained model first resizes the

image so that the longer side has 224 pixels, followed by a 224×224 center crop. Then, the model encodes each frame into a 512-dimensional vector.

Audiotag To extract audio features, we use a model pretrained for audio event tagging (Kong et al., 2020). This model processes 3-second audio chunks and outputs a probability vector for 527 different labels. We extract the activations before the classification layer, resulting in 128-dimensional vectors for each audio chunk.

Music Since cinematic trailers often feature significant soundtracks, it is essential to investigate the musical aspect further. To capture this, we extract a musical feature using a model pretrained for music genre classification (Choi et al., 2016). This model processes 22-second audio chunks and outputs a probability vector for 50 different labels and 64-dimensional activations for each audio chunk.

We used the Audiotag and Music models only for genre classification and later replaced them with the following improved and unified audio analysis model for emotion classification.

BEATs Bidirectional Encoder representation from Audio Transformers (BEATs) is an audio classification model that combines an acoustic tokenizer and a classifier, which are trained iteratively (S. Chen et al., 2023). This model is capable of classifying both audio events and music genres.

Face detector Human faces are detected and extracted using the model from the Ultralytics group[1], which is based on YOLO (You Only Look Once) (Bochkovskiy et al., 2020). YOLO is a real-time object detection algorithm that divides an image into a grid and predicts bounding boxes and class probabilities for each grid cell using convolutional neural networks. This model was later included for emotion classification and was not used for genre classification.

Expression classifier The cropped human faces are then fed into a facial emotion classifier[2]. This model, a Vision Transformer (ViT) (Dosovitskiy et al., 2021), was trained on the FER-2013 (Facial Emotion Recognition) dataset (Barsoum et al., 2016). The model takes a facial image and predicts the facial expression as angry, disgusted, fearful, happy, sad, surprised, or neutral. This model was later included for emotion classification and was not used for genre classification.

Automatic Speech Recognition (ASR) OpenAI's Whisper model processes audio input, detects speech patterns, and outputs text, as done in the genre classification task (Radford et al., 2023). We use the larger variant of the model which also translates non-English text into English.

[1] https://github.com/ultralytics/ultralytics

[2] https://huggingface.co/trpakov/vit-face-expression

Optical Character Recognition (OCR) We also performed optical character recognition (OCR) on each frame using the PaddleOCR model[3]. We additionally used a pretrained spell correction model on the produced output[4]. The overall output of this stage is in text format. We improve the OCR output by incorporating the following additional post-processing models.

Language identification and translation We translate non-English OCR text into English as the final text encoding models are trained on English text. We first use a language identification model[5] to detect if translation is necessary. This model, based on XLM-RoBERTa (Conneau et al., 2020), is trained on multiple language identification datasets (Conneau et al., 2018; Keung et al., 2020; May, 2021).

The non-English OCR text is then translated into English using Facebook's NLLB-200 (No Language Left Behind) model (Costa-jussà et al., 2022). This model uses a combination of transformer (Vaswani et al., 2017) and Sparsely Gated Mixture of Experts (Almahairi et al., 2016) layers, trained on internet-sourced text data, to translate between 200 languages. While the NLLB-200 model requires the source language to be specified, the language identification model automatically detects it. As a large language model, NLLB-200 can translate long texts and yield error-free outputs even if the input contains spelling errors. However, for English OCR output, we use the following correction models.

Spell correction The Symspell package provides tools for spell checking[6]. Among these tools, the word segmenter separates words in sentences where spaces are missing, which is useful for post-processing OCR outputs that lack spaces. We also use a spellcheck model to correct potential typos[7]. This is a T5 transformer language model (Raffel et al., 2020) trained on a dataset containing synthetic spelling errors, allowing it to correct any English text. After obtaining the final text output from ASR and OCR, we encode them to obtain feature vectors, using pretrained language models.

DistilBERT For genre classification, we encode the output text of the OCR and ASR models using a pretrained language model, specifically DistilBERT (Sanh et al., 2019), which is a condensed and compressed variant of the BERT (Bidirectional Encoder Representations from Transformers) model (Devlin et al., 2019), achieved through knowledge distillation (Bucila et al., 2006; Hinton et al., 2015). DistilBERT utilizes fewer layers than BERT and learns from BERT's outputs to mimic its behavior. This model converts the input text into tokens, including words and sub-words, and encodes each token as a 768-dimensional vector.

[3]https://paddlepaddle.github.io/PaddleOCR/main/en
[4]https://huggingface.co/oliverguhr/spelling-correction-english-base
[5]https://huggingface.co/papluca/xlm-roberta-base-language-detection
[6]https://github.com/wolfgarbe/SymSpell
[7]https://huggingface.co/ai-forever/T5-large-spell

Text sentiment classifier For emotion classification, we use another language model that is trained for sentiment analysis. We use a RoBERTa language model (Y. Liu et al., 2019) that was trained on the TweetEval sentiment classification benchmark (Camacho-Collados et al., 2022)[8]. This model can predict the sentiment of a given text as positive, negative, or neutral. As with the previous features, we utilize the activations before the final classification layer.

After introducing our pretrained features, we now discuss the models used to process these features and make final predictions for the emotion and genre classification tasks.

5.2.2 Emotion classification

In this task, we aim to classify input videos into emotion categories. We name our model VEMOCLAP, which stands for **V**ideo **EMO**tion **CLA**ssifier using **P**retrained features. We begin by describing the dataset used and our approach for extracting pretrained features from it.

5.2.2.1 Dataset and feature extraction

We use the Ekman-6 dataset, as it is one of the largest emotion-labeled datasets that includes arbitrary user-generated videos (B. Xu et al., 2018). We avoid using datasets that feature more specific types of videos, such as those focused on human faces or social interactions, because we want our final video-based music generator to be applicable to any type of video. The Ekman-6 dataset contains 1,637 short videos, split equally for training and testing. Each video is labeled with one of the six basic emotions (Ekman, 1971): anger, disgust, fear, joy, sadness, and surprise.

We extract a fixed number, n, of frames from each video, along with the entire audio, which is resampled at 16 kHz and converted to mono. The features from the facial expression classifier and CLIP are sequences of vectors, as their inputs are sequences of frames. If multiple faces are detected in a single video frame, we average the features of the two largest faces. Similarly, the BEATs model processes sequences of 3-second audio chunks. We extract n audio chunks to match the number of video frames. While CLIP and BEATs produce n output vectors, the facial expression classifier may produce fewer vectors, depending on whether faces are present in each frame.

The ASR model generates a single block of text from the entire audio. If the source language is not English, it automatically translates the output text into English. The OCR model generates blocks of text for each video frame. The language identification model processes each block, and the text is translated into English if the identified language is not English. If the language is already in English, the text is passed through the word segmenter and then the spell corrector. The resulting text blocks from each frame are concatenated to form a single block of text. The texts resulting from ASR and OCR are fed into the sentiment classifier separately. Since the sentiment classifier predicts a single label for any length of text, a single vector is extracted as the feature.

After feature extraction, for a single video, we are left with n CLIP, n BEATs, $k \leq n$ facial expression, 1 OCR sentiment, and 1 ASR sentiment feature. Fusing these pretrained features

[8] https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment

presents several challenges. First, since they are extracted using different pretrained models, the lengths (dimensionalities) of the feature vectors vary. Second, when the feature vectors form a sequence, as in the case of facial expressions, CLIP, and BEATs, their temporal lengths can also differ. Finally, since these features belong to different modalities, the content of the feature vectors can be vastly different. We now present the method we use to fuse the pretrained features and produce the final emotion predictions.

5.2.2.2 Classification method

To handle the value range difference of the multimodal features, we first perform min-max normalization on each feature using statistics extracted from the collection of pretrained features. To handle differing dimensionalities, all sequential input features are first projected to queries, keys, and values with a common dimensionality (Vaswani et al., 2017). Next, the cross-attention modules (Bahdanau et al., 2015) exploit correspondence between pairs of sequential features. The attention modules also include dropout and layer normalization and can handle a pair of sequences with different temporal lengths. As done in classification tasks, the attention outputs are averaged along the temporal dimension, yielding a single vector. Since the OCR and ASR sentiment features are already single vectors, each modality is represented by a single vector after the attention modules. We then concatenate all five feature vectors along the channel dimension, resulting in a single vector representing the entire video. Finally, this vector is fed into a linear layer followed by a softmax layer, which outputs a probability for each emotion.

Our model is shown in Figure 5.1. Blocks with rounded and dashed outlines represent trained modules. The models with parentheses are used conditionally. Other blocks are pretrained feature extractors and are used in inference mode. Q, K, and V represent query, key, and value projections.

5.2.2.3 Implementation details and hyperparameters

We initially extract video frames at 1 frame per second, using $n = 16$ video frames and corresponding audio chunks as input to our model. Due to the limited size of the Ekman-6 dataset, we adopt this small, fixed input length to reduce the risk of overfitting. In the subsequent genre classification task, where more data is available, we lift this constraint.

During training, we select the n video frames and audio chunks from random locations for data augmentation. During testing and inference, we extract them at equidistant intervals to ensure comprehensive temporal representation. We report classification performance using the provided training and testing splits, which included 819 and 818 videos, respectively.

We use cross-entropy loss, a batch size of 32, a dropout rate of 0.5, and Adam optimizer with a learning rate of 1e-5 (Kingma & Ba, 2015). Attention modules have 4 heads and a dimensionality of 512. The model has around 11M trainable parameters. We used 10% of the training split as the validation split and stopped training when validation accuracy started to drop.

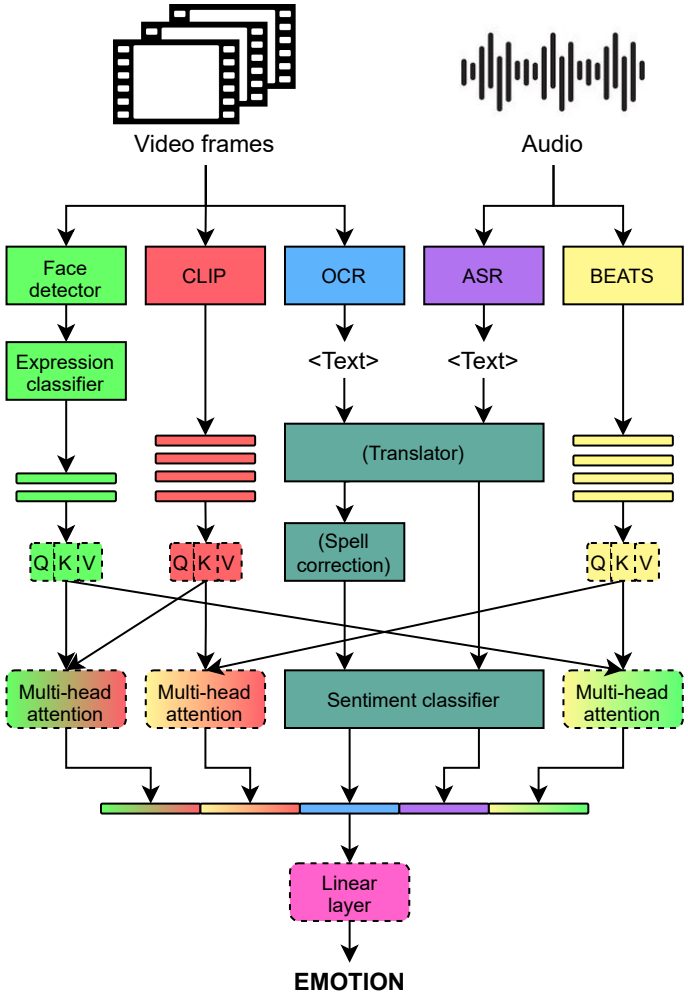


Figure 5.1: Video emotion classification pipeline.

5.2.2.4 Dataset cleaning

Our initial examination of the Ekman- 6 dataset revealed several problematic samples. While we report classification results on the unedited Ekman-6 dataset, we also clean it to train the model used in our video-based music generator later in Chapter 6. The Ekman-6 dataset was created by scraping the web for videos using search keywords that matched not only the categorized emotion but also related terms. We view each video to detect the problematic samples. After inspecting their file names, we identify the following problematic search keywords for each emotion class, which are underlined.

Anger: A single person being annoying, with no other person present to be annoyed or angry.

Disgust: Flashing lights or rapid camera movement, presumably to induce dizzyness or nausea. It also includes videos related to boredom and loathing.

Fear: Counter-terrorism, underwater footage, 9/11 terrorism attack aftermath, and suspect apprehension.

Joy: Joyride (driving a car), the music "Ode to Joy", and people named Joy.

Sadness: pensive

Surprise: distraction, and people performing impressive feats labeled as astonishing.

We identify and remove 128 and 130 problematic videos from the training and testing splits, respectively. Using the cleaned data, the classification accuracy increases by 2.6%. However, we exclude this result from our comparison with the state-of-the-art because data cleaning alters the test split's content, affecting the comparison's fairness.

5.2.2.5 Inference web application

As a complementary work, we present an open-source web application for performing inference on user-provided videos, designed to support both the general public and the research community[9]. Hosted on Google Colab, it offers free GPU runtime. Users can upload their own videos, provide a YouTube link, or use sample videos provided within the application. The application is self-contained and ready to use, requiring no setup from the user. The process is streamlined into 5 steps, with only 5 mouse clicks needed to obtain the results. After connecting to a GPU runtime, users should follow these steps:

Step 1: Automatically download and extract the codebase, and install the required Python libraries. This step takes approximately 2 minutes.

Step 2: Download and build the feature extractor models and the classifier model. As these models are deep neural networks, this step takes about 3 minutes. Note that Steps 1 and 2 only need to be completed once, even if classifying multiple videos.

Step 3: Select how to load the input video. The options are "Sample video", "YouTube link", and "Upload video".

Step 4: Depending on the choice from step 3, the user then selects the specific video. Steps 3 and 4 take only a few seconds to complete.

Step 5: Extract the frames and audio from the input video, run the pretrained feature extractors, and finally run the emotion classifier. The outputs include text from automatic speech recognition (ASR) with its sentiment, text from optical character recognition (OCR) with its sentiment, and predictions from the BEATs audio classifier. Additionally, a sample frame is displayed showing detected faces with predicted emotional expressions, detected OCR boxes, and a caption generated by CLIPCap (Mokady et al., 2021). Note that this sample frame is for demonstration purposes, while all n frames are used for the final emotion classification. For a 60-second video, this step takes approximately 30 seconds.

5.2.2.6 Results and discussion

In Table 5.1, we present the quantitative performance of our model on the Ekman-6 dataset using the provided training and testing splits, showing that our method outperforms the state-of-the-art by 4.3%.

[9] https://www.serkansulun.com/app

Table 5.1: Classification accuracies compared to the state-of-the-art on the Ekman-6 dataset.

Method	Accuracy (%)
ITE (B. Xu et al., 2018)	51.20
CFN (C. Chen et al., 2016)	51.80
MART (Z. Zhang et al., 2024)	53.17
VAANet (S. Zhao et al., 2020)	55.30
CTEN (Z. Zhang et al., 2023)	58.20
KeyFrame (Wei et al., 2021)	59.51
LRCANet (Yi et al., 2024)	59.78
FAEIL (H. Zhang & Xu, 2023)	60.44
TAM (Pan et al., 2022)	61.00
VEMOCLAP (Ours)	**65.28**

Figure 5.2 shows the confusion matrix for our classification results on the test split. Disgust has the lowest classification accuracy among the six emotions. This may be because, unlike the others, disgust is highly influenced by social conditioning and shaped by cultural moral values (Haidt et al., 1997). 20% of videos labeled as sadness are misclassified as fear, likely due to the shared negative emotional tone of both categories. Similarly, 16% of videos labeled as joy are misclassified as surprise. Based on our manual inspection, this may be because most videos in the surprise category convey a positive affect—often depicting individuals who are not only surprised but also joyful.

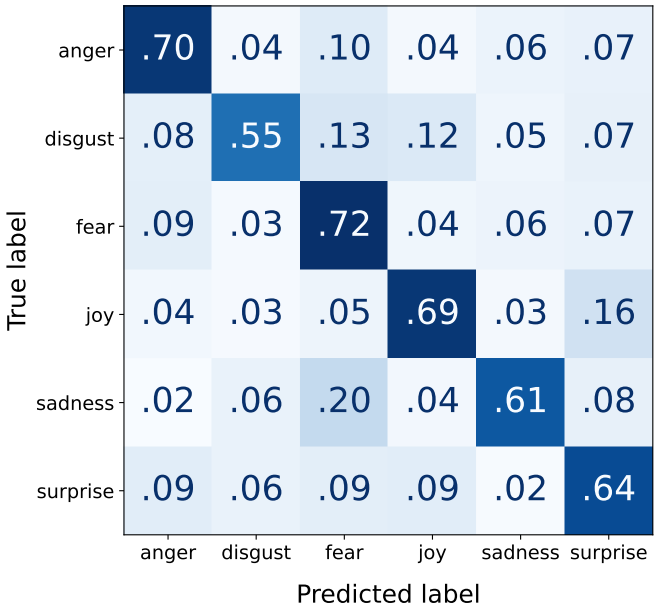


Figure 5.2: Confusion matrix with values normalized over true labels on the test split of Ekman-6 dataset.

To the best of our knowledge, our inference web application is the first click-and-run software for video emotion classification. The trained VEMOCLAP is also incorporated into our video-based music generation pipeline as we will explain in Chapter 6.

5.2.3 Genre classification

We continue with another semantic task, namely genre classification. In this task, we exploit a large-scale cinematic dataset to challenge the limitations on number of input frames.

5.2.3.1 Dataset and feature extraction

We use the MovieNet cinematic dataset, which includes metadata for 375k movies and YouTube trailer links for 33k of them, making it the largest labeled trailer dataset available. Importantly, this dataset is multi-label, indicating that an individual video can be associated with multiple genres.

We extract CLIP, OCR, ASR, Audiotag, and Music features as illustrated in Figure 5.3. Unlike the emotion classification pipeline, we process the text features using the general-purpose language model DistilBERT, which produces a sequence of feature vectors corresponding to each token (word or subword).

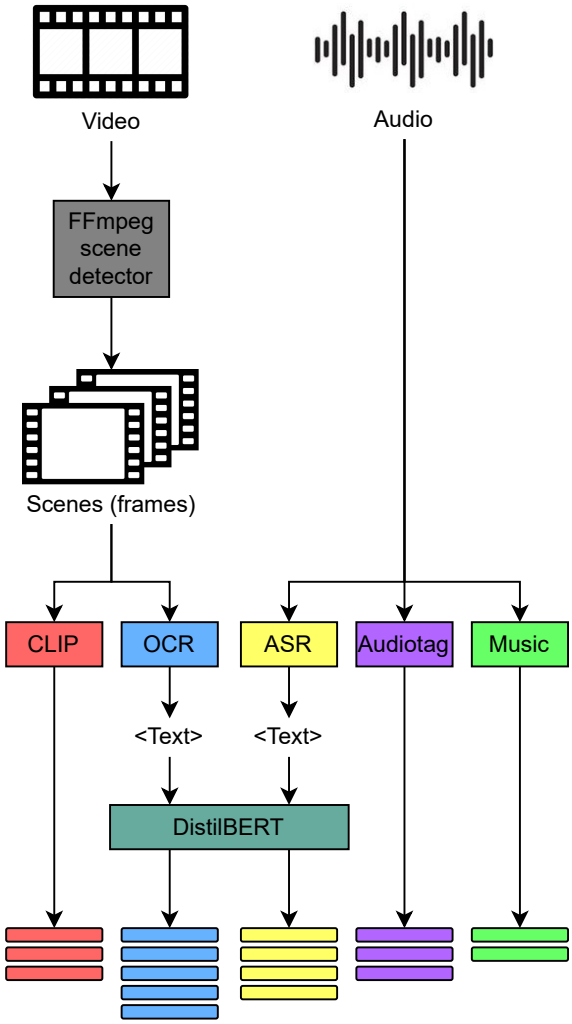


Figure 5.3: Video feature extraction pipeline for genre classification.

5.2.3.2 Classification methods

We build multiple models to take the previously extracted features of a given video as input and predict the genre of the video. Combining different features is challenging, especially since the encoded vectors have different lengths in both channel and temporal dimensions. For example, considering the audio event and music features, the lengths of each vector are 128 and 64, respectively. Furthermore, since the audio event and music networks operate on chunks of audio with lengths of 3 and 22 seconds respectively, for the same video, there are more embedded vectors for the former. And finally, the number of vectors for the same feature differs between different videos. We address these issues using different solutions and different classification models.

We build and train three distinct models to fuse and classify pretrained features: *a multi-layer perceptron (MLP)*; the *single-transformer* model that integrates features across all modalities; and the *multi-transformer* model, where individual transformers handle features from specific modalities. As a baseline, we also implement a simpler model that does not utilize diverse pretrained features and instead uses two branches to independently process raw video frames and audio.

The final layer of all our models is a fully-connected (FC) layer with a size of 21, outputting probabilities belonging to 21 different genres. This layer is followed by a sigmoid layer to make sure each output is a probability between 0 and 1.

MLP We first implement a simple MLP classifier, mirroring the state-of-the-art approach used by Q. Huang et al. (2020). While their method processes raw video and audio from a few short segments, we instead use pretrained features extracted from the entire video. Since MLPs require fixed-length input, we follow a similar strategy by averaging the feature vectors from each modality over time. These averaged vectors are then concatenated and passed to the MLP. Our architecture is illustrated in Figure 5.4.

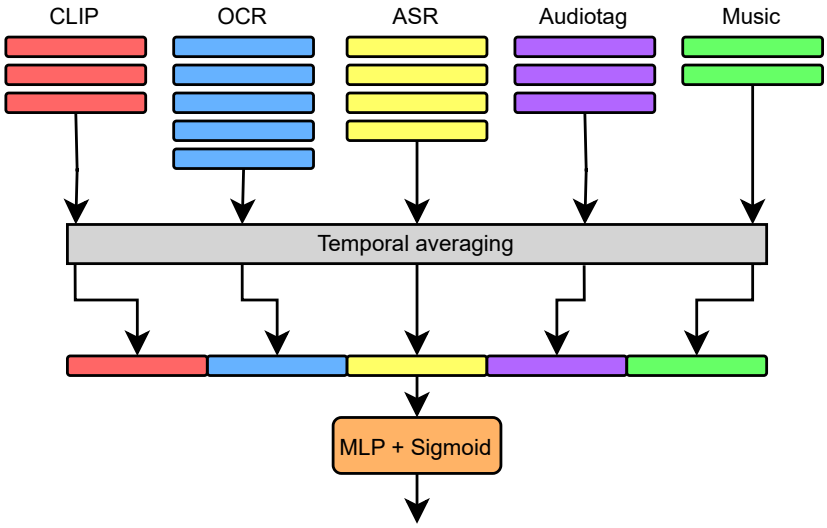


Figure 5.4: Our MLP model.

In our two other models namely single-transformer and multi-transformer, we optionally use the temporal averaging for text features such as OCR and ASR. While non-textual features, namely CLIP, Audiotag, and Music are obtained from the activations *before* the final prediction layer, the OCR and ASR features stem from running the DistilBERT model on the predicted text. Here, any error in the predicted text is propagated into the DistilBERT model, potentially corrupting the output features. In order to reduce the resulting noise, we experiment with averaging the textual features, namely OCR and ASR, along the temporal dimension.

Single-transformer To mitigate the significant information loss caused by averaging features along the temporal dimension, we employ a transformer model to capture both short- and long-term correspondences within the video sequence. We use the transformer as a sequence classifier by prepending the sequences with a special learnable vector, known as the `<CLS>` vector. While the transformer generates output vectors for each element in the input sequence, in classification tasks, only the output vector corresponding to the `<CLS>` vector is passed to the next layer, and the others are discarded (Devlin et al., 2019).

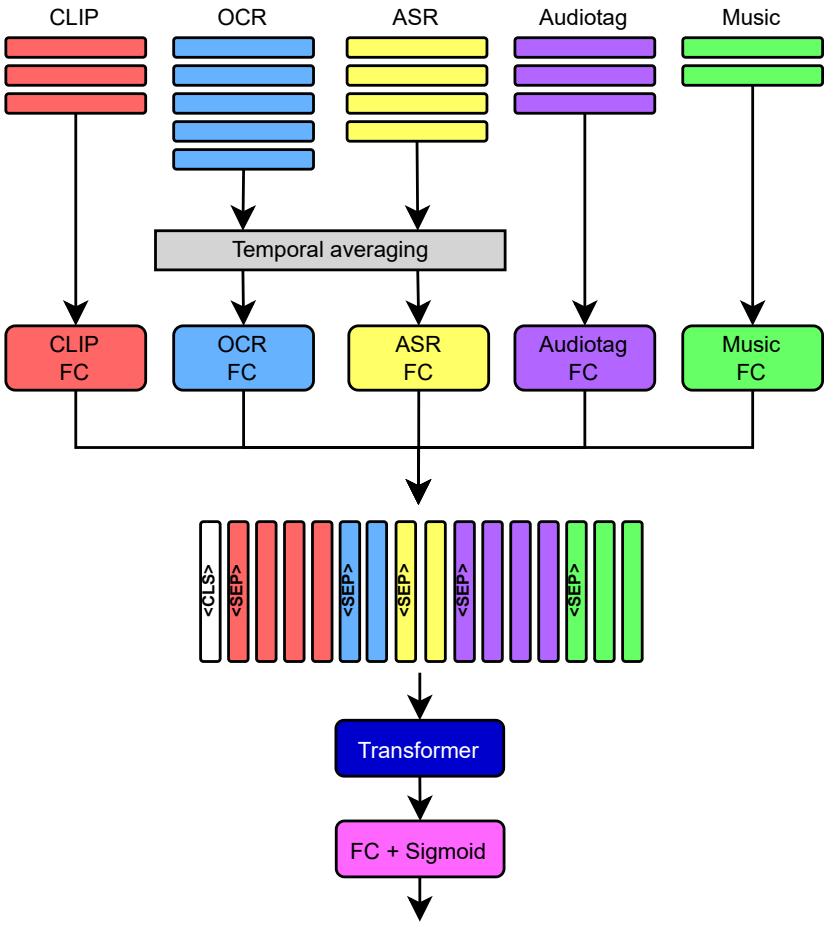


Figure 5.5: Our single-transformer model.

While the transformer can handle sequences with varying lengths, the vectors in the sequence still need to have equal sizes along the channel dimension. To ensure that, we use fully-connected layers to transform feature vectors from different modalities into the same size. We note that this layer is applied to each vector in the sequence individually, hence not changing the number of vectors in its input sequence. Next, we concatenate all the vectors along the time dimension before feeding them into a single transformer. This approach aims at exploiting the correspondences between different features at the temporal level by processing all of them as a single sequence. However, a trade-off is the complexity of handling different modalities with a single transformer block using a single set of weights.

To make sure the transformer can distinguish feature vectors from different modalities, we use separately learned positional embeddings for sequences from each modality (B. Wang et al., 2020). We additionally make use of separator (<SEP>) vectors, namely, an encoded version of the <SEP> token (Devlin et al., 2019). These are learned vectors that are specific for each modality, prepended to each sequence. Finally, as in almost all classifier transformers, we prepend the input sequence to the transformer with a learnable classifying (<CLS>) vector, namely, an encoded version of the <CLS> token (Devlin et al., 2019). The output vector corresponding to this classifying vector, i.e. the first vector, is considered as the prediction of the transformer and fed into the final fully-connected layer, while the remaining vectors of the transformer's output sequence are discarded. The overall model is shown in Figure 5.5.

Multi-transformer Given the hypothesis that using a single transformer to process features from multiple modalities can be inefficient due to the increased complexity of the input, we devise a final model that incorporates separate transformer models for each modality, as shown in Figure 5.6. The inputs to all transformers are prepended with <CLS> token vectors, and the corresponding output vectors are concatenated along the channel dimension before being fed into a fully connected layer to obtain the final probabilities. This approach leverages potential correspondences between different features at the global level, though not at the temporal level.

Vision transformer baseline model To evaluate our approach of using multimodal pretrained features, we also implement a strong baseline for comparison. Unlike our main models, this baseline doesn't incorporate a wide range of pretrained features, but it still leverages powerful pretrained networks along with additional trainable layers. It operates on sequences of 2-dimensional raw visual frames and audio spectrograms, in contrast to the previously introduced models that work with sequences of 1-dimensional pretrained features.

Our baseline model uses the Vision Transformer (ViT) to divide each video frame into patches, encode them, and process the resulting sequence of vectors with a standard transformer architecture (Dosovitskiy et al., 2021). For audio, it employs the Audio Spectrogram Transformer (AST), a variant of ViT designed to operate directly on audio spectrograms (Gong et al., 2021). Despite being purely attention-based, without any convolutional layers, ViT outperforms convolutional neural networks, achieving state-of-the-art results in image (Dosovitskiy et al., 2021), video (Arnab

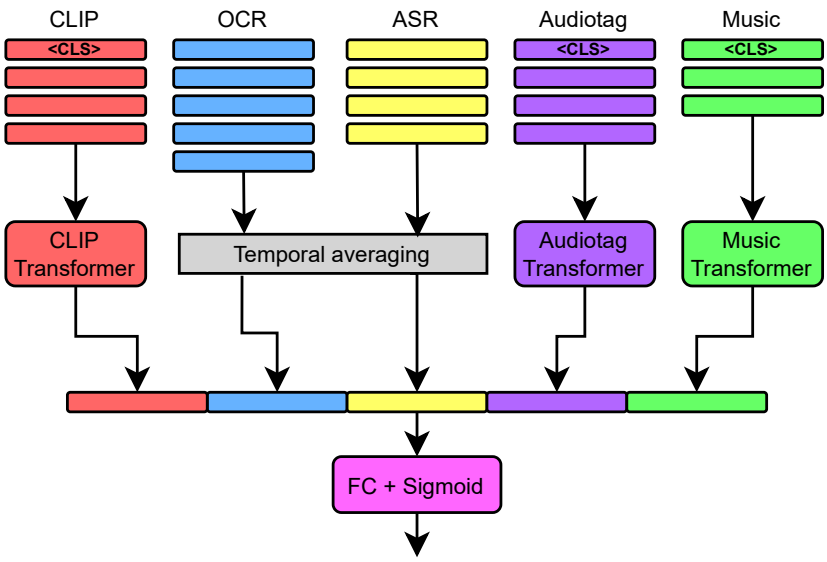


Figure 5.6: Our multi-transformer model.

et al., 2021), and audio (Gong et al., 2021) classification.

While our primary objective is video classification, directly applying the video vision transformer (ViViT) (Arnab et al., 2021) to our task is not a suitable approach for two main reasons. First, ViViT does not incorporate audio, which is a crucial aspect of our task. Second, our dataset contains discontinuous frames that depict different scenes with no visual continuity between them. As a result, it is more appropriate to handle these frames individually rather than concatenating patches from different frames, as ViViT does.

The ViT and AST models are pretrained on the ImageNet (Deng et al., 2009) and AudioSet (Gemmeke et al., 2017) datasets, which are large-scale benchmarks for image and audio classification, respectively. In our preliminary experiments, we attempted to train the entire model, including fine-tuning ViT and AST, but observed severe overfitting that could not be alleviated by standard regularization techniques such as dropout (Srivastava et al., 2014). To avoid this, we keep the parameters of the ViT and AST frozen and replace their output layers with trainable MLPs, naming them image-MLP and audio-MLP.

The pretrained AST is designed to work with audio spectrograms organized into 10-second segments, which we refer to as "spectrogram frames." To distinguish them, we use the term "image frames" to describe the visual scenes in our context. Since the trailers in our dataset are longer than 10 seconds, we split the full audio spectrogram into 10-second chunks, with a 50% overlap. Similarly, the ViT works with individual image frames, while our trailers consist of multiple frames. This leads to an output sequence of vectors, where each vector corresponds to an individual image or audio frame. To process these sequences and obtain the final predictions, we produce a fusion module. To this end, we use standard transformer classifiers for both image and audio sequences, yielding a single vector for each modality. Finally, we concatenate these two vectors

and process them using a linear layer and a sigmoid layer to obtain the final label predictions.

Even when the parameters of the pretrained ViT and AST models are kept frozen, training the full model in an end-to-end fashion leads to overfitting. This is due to the relatively small size of our training dataset, which contains only 33k samples, while the original authors of ViT and AST train their models on extensive datasets such as Imagenet with 21M samples (Deng et al., 2009) and AudioSet with 2M samples (Gemmeke et al., 2017), respectively. To address this overfitting issue we employ a two-stage training approach. First, we train the image- and audio-MLPs separately on individual frames. During this phase, the target is the label of the video to which the frames belong. Next, we freeze the parameters of the MLPs and train the fusion module using complete videos. The baseline model and its training scheme are illustrated in Figure 5.7.

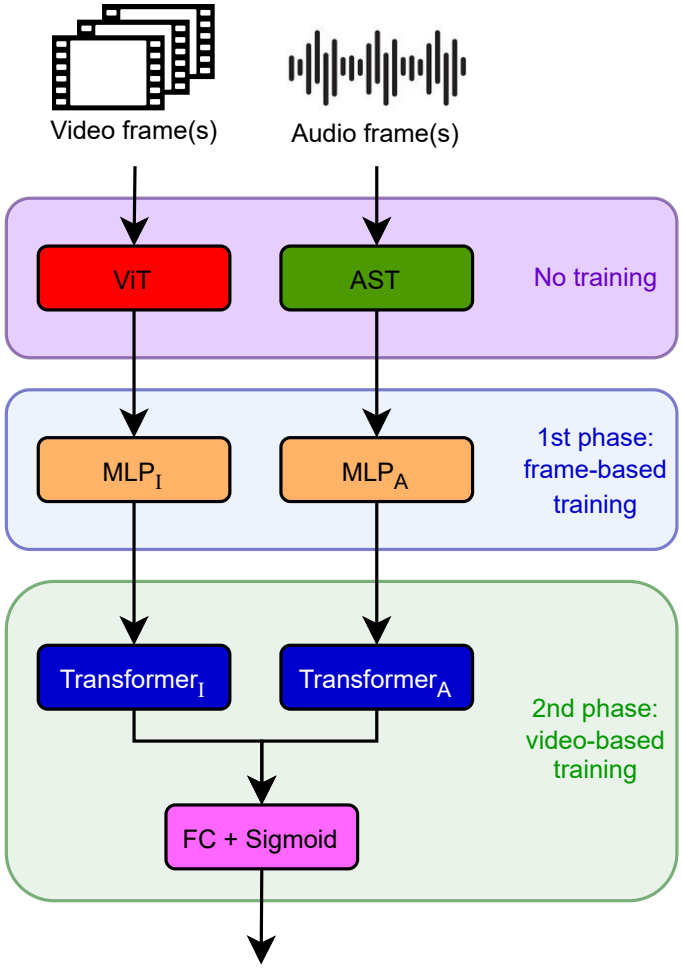


Figure 5.7: Our baseline model. The subscripts "I" and "A" refer to image and audio, respectively.

5.2.3.3 Implementation details and hyperparameters

We apply a filtering process to exclude extremely long and short videos from our dataset. Specifically, we compute the quartiles of video durations and exclude samples that were outside the inner

fence, i.e., durations shorter than $Q1 - 1.5 \times IQR$ (corresponding to 19.6 seconds) and longer than $Q3 + 1.5 \times IQR$ (corresponding to 214.4 seconds). This filtering yields 26412 videos from the original set of 32647. Moreover, following the approach in the MovieNet paper (Q. Huang et al., 2020), we limit the labels to the 21 most frequent genres, namely, action, adventure, animation, biography, comedy, crime, documentary, drama, family, fantasy, history, horror, music, musical, mystery, romance, sci-fi, sport, thriller, war, and western.

To ensure an unbiased split of the dataset, we arranged the videos alphabetically by their YouTube IDs—given that these IDs are generated randomly. Following the methodology in the original MovieNet paper, we divide the dataset into training, validation, and testing splits with ratios of 0.7, 0.1, and 0.2, corresponding to 18488, 2641, and 5283 samples respectively (Q. Huang et al., 2020). Hyperparameters are determined using a grid search, optimizing for the highest mean average precision on the validation split. The test split is utilized solely for reporting the final results.

While a straightforward way to balance precision and recall is to change the decision threshold, this method does not allow for comparison against the works in the literature where most report their results while setting a fixed decision threshold at 0.5 (Q. Huang et al., 2020). Alternatively, during training, we can balance precision and recall by applying a constant weight to the loss associated with positive labels. For a single sample, the weighted binary cross-entropy loss becomes:

$$Loss = -\frac{1}{C}\sum_{c=1}^{C} w \cdot y_c \cdot \log(p_c) + (1 - y_c) \cdot \log(1 - p_c)$$

Here, w represents the weight for positive labels and C is the total number of classes. To align precision and recall values with those observed in MovieNet, we apply a weight of 0.25.

Both the transformer architecture and the averaging module of the MLP architecture can handle sequences of varying lengths. However, during training, we use fixed-length inputs to facilitate minibatch processing. Longer sequences are truncated to the desired length, while shorter ones are padded with zero vectors. The length of feature vectors for each pretrained modality is determined through exploratory data analysis, specifically using box plots to analyze sequence lengths across all samples. The maximum sequence length is set to the upper adjacent value (upper whisker of the box plot), which is 1.5 times the interquartile range above the third quartile. As a result, the sequence lengths are defined as 216 for CLIP, 64 for OCR, 86 for ASR, 140 for Audiotag, and 18 for Musicnet. When using a single-transformer model, the concatenated sequence length totals 524. It is important to note that during inference on individual samples, our models can handle inputs of varying lengths without the need for padding or truncating.

For the MLP model, there is 1 layer, a model dimension of 256, and a total of 57k parameters. The single-transformer model consists of 2 layers, 8 attention heads, and a model dimension of 256, with a total of 8.56M parameters. The multi-transformer model features individual transformers, each with 1 layer, 8 attention heads, and a model dimension of 128, totaling 6.98M

parameters. Considering the baseline model, both the image-MLP and audio-MLP have 2 layers and a dimensionality of 768. The fusing transformers each have 1 layer, 4 attention heads, and a dimensionality of 768. All transformers are trained with gradient clipping at a norm of 1. Models are trained using the Adam optimizer (Kingma & Ba, 2015), with a learning rate of 1e-5, a dropout rate of 0.5, and a batch size of 32. The models are implemented using the PyTorch library (Paszke et al., 2019) and trained on a single NVIDIA Quadro RTX 6000 GPU with 24 GB of memory.

5.2.3.4 Results and discussion

We first compare the performance of our models against the baseline model and the models presented in the MovieNet paper (Q. Huang et al., 2020). Inference is performed on the test set one sample (video) at a time, using the full duration of each video without padding or truncating the sequences. Following the literature, we report macro-averaged precision, recall, and mean average precision (mAP) values, averaged over individual labels (genres).

Table 5.2 shows the overall performance of our models compared to models in the literature. In this table, P, R, and mAP denote precision, recall, and mean average precision respectively. For precision and recall, 0.5 is used as the decision threshold. The results for the state-of-the-art models are taken from the original MovieNet paper (Q. Huang et al., 2020). For clarity, the full names of the abbreviated models are: TSN (Temporal Segment Network) (L. Wang et al., 2016), I3D (Two-Stream Inflated 3D ConvNets) (Carreira & Zisserman, 2017), and TRN (Temporal Relation Network) (B. Zhou, Andonian, et al., 2018).

Our baseline model outperforms the models in the literature in terms of recall and mean average precision. Models using pretrained features outperform all other models across all metrics. Our best-performing model, the multi-transformer, significantly improves the state of the art. It outperforms all other models in every metric except precision, where it performs marginally worse than our MLP model.

Table 5.2: Performance of our models against the state of the art.

Models	P@0.5	R@0.5	mAP
TSN	78.31	17.95	43.70
I3D	69.58	16.54	35.79
TRN	77.63	21.74	45.23
MovieNet	79.74	24.97	46.88
Baseline	73.78	32.17	59.73
MLP	**82.05**	33.51	63.16
Single-transformer	81.00	37.11	65.09
Multi-transformer	82.00	**38.33**	**66.02**

In Table 5.3 we present a detailed analysis of the results across all genres. For conciseness, we only show the results for the best model in the literature MovieNet (MN) (Q. Huang et al., 2020), our baseline (BL) model, and our best performing model multi-transformer (MT). P, R, and mAP

denote precision, recall, and mean average precision respectively. For precision and recall, 0.5 is used as the decision threshold.

Table 5.3: Results per genre and the macro averages. MN: MovieNet (Q. Huang et al., 2020), BL: baseline (ours), MT: multi-transformer (ours).

	P@0.5			R@0.5			mAP		
	MN	BL	MT	MN	BL	MT	MN	BL	MT
Action	73.96	86.59	**87.37**	22.21	40.48	**48.12**	54.60	75.64	**79.20**
Adventure	75.24	74.71	**77.51**	24.72	22.48	**28.67**	53.06	56.76	**60.93**
Animation	93.16	**96.38**	96.19	74.09	86.59	**92.28**	86.45	94.65	**96.18**
Biography	**100.00**	53.85	70.00	0.04	2.86	**5.71**	9.13	24.29	**36.68**
Comedy	68.61	88.55	**90.39**	48.65	**57.94**	57.05	68.81	85.47	**87.84**
Crime	74.12	72.00	**80.22**	**39.30**	8.33	22.53	49.25	50.81	**60.25**
Documentary	85.49	91.30	**92.37**	4.79	70.87	**81.71**	21.03	90.29	**94.64**
Drama	71.16	85.70	**89.40**	**79.42**	49.16	55.06	79.95	83.08	**86.77**
Family	82.55	83.08	**86.58**	27.11	40.14	**48.08**	52.19	67.09	**73.60**
Fantasy	69.83	79.37	**87.88**	13.51	12.02	**20.91**	39.12	48.78	**59.59**
History	82.90	63.16	**90.00**	**12.52**	6.49	9.73	34.41	33.63	**41.05**
Horror	70.03	86.49	**88.38**	8.76	51.04	**55.82**	35.51	79.48	**84.24**
Music	**89.04**	66.67	68.06	27.24	20.48	**29.52**	47.13	42.17	**48.36**
Musical	73.58	70.59	**75.00**	4.45	11.21	**14.02**	22.88	32.86	**41.27**
Mystery	**76.42**	0.00	57.14	**7.76**	0.00	1.20	**39.70**	25.28	30.68
Romance	71.93	76.47	**82.32**	14.02	13.65	**15.75**	49.27	51.16	**58.59**
Sci-Fi	81.35	72.09	**83.60**	14.51	29.67	**37.80**	44.14	54.11	**68.00**
Sport	**94.97**	69.51	75.86	21.99	42.86	**49.62**	39.59	60.44	**66.80**
Thriller	64.98	**79.35**	78.17	14.50	28.79	**37.82**	49.80	69.54	**71.76**
War	**86.27**	64.71	75.86	12.80	20.89	**27.85**	34.41	47.66	**56.40**
Western	88.89	88.89	**89.80**	51.93	59.70	**65.67**	73.99	81.14	**83.53**
AVERAGE	79.74	73.78	**82.00**	24.97	32.17	**38.33**	46.88	59.73	**66.02**

Out of all 21 genres, our model outperforms the other models in 14 of them in terms of precision, 16 of them in terms of recall, and 20 of them in terms of mean average precision. Outlier values such as very low recall for genres such as Biography, History, and Mystery can be attributed to the inherent imbalance within the MovieNet dataset (Q. Huang et al., 2020). We deliberately avoided balancing the data before training to maintain a fair comparison with the classification model presented in the MovieNet paper. We also believe that this dataset closely reflects the real-world distribution of cinematic genres, accurately portraying the relative rarity of genres such as Biography, History, and Mystery.

In Table 5.4 we present an ablation study using our best-performing model multi-transformer, demonstrating the gain in terms of mean average precision along with the increase in runtime, due to the addition of each feature while comparing against our baseline model. The first row shows the performance of the baseline model. Asterisk ($*$) indicates averaging the features over the sequence (temporal) dimension. Runtime is defined as the average duration in seconds, to process a single video, both extracting its pretrained features and classifying it, during inference.

Table 5.4: The effect of inclusion of pretrained features on mean average precision and runtime.

CLIP	Music	Audiotag	OCR	ASR	mAP	Runtime (s.)
					59.73	7.08
✓					64.73	5.32
✓	✓				65.17	5.76
✓	✓	✓			65.31	5.95
✓	✓	✓	✓		63.33	25.85
✓	✓	✓	*		65.46	31.52
✓	✓	✓	✓	✓	64.66	31.44
✓	✓	✓	*	*	**66.02**	33.59

Although the inclusion of textual features such as OCR and ASR initially seems to hurt the performance, possibly due to the text prediction errors, incorporating their averaged version along the temporal dimension reduces the noise and does improve mean average precision. Incorporating textual features also significantly increases runtime. This is primarily because models such as CLIP, Music, and Audiotag are designed to predict a single embedding vector or label, while OCR and ASR models predict text sequences, which can be viewed as predicting many labels corresponding to each word and subword, in an autoregressive way. But most importantly, the settings in which the textual features are excluded yield a better classification performance and faster runtime compared to the baseline. Furthermore, our classification models can seamlessly incorporate features that result from newer and potentially more efficient ASR and OCR models that can be developed in the future, further reducing the overall runtime.

Finally in Figure 5.8 we attempt to further explain the reasons behind the success of our models. The models in the literature extract a small and fixed number of frames at random locations from each video for classification. The MovieNet baseline (Q. Huang et al., 2020) only uses 8 scenes. They furthermore use MLPs, which cannot process sequences, hence the features from these small number of frames need further averaging along the temporal dimension, causing additional loss of information. Using a simplified experiment, we show the importance of the number of frames fed into the model. Using only the CLIP features, we employ 8, 16, 32, 64, 128 and 256 frames obtained from random locations in each video. We only use the CLIP features since the number of feature vectors differs amongst the pretrained features, and finding an exact number of features that would correspond to the specific number of CLIP features is challenging.

As we use a single input modality, both our single-transformer and multi-transformer architectures defaults to a standard transfomer classifier. We therefore process the CLIP feature sequences using an MLP and a transformer. Regarding the MLP, the features from different frames are averaged along the temporal dimension as in (Q. Huang et al., 2020). On the contrary, the transformer model is capable of processing sequences hence we feed the features from each frame *as is*.

In Figure 5.8, we can see that using a higher number of frames consistently improves the performance. Furthermore, as the number of frames increases, the superiority of the transformer over the MLP becomes apparent, due to its ability to process long sequences seamlessly.

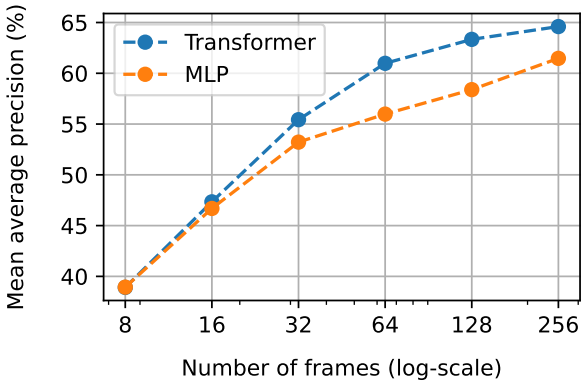


Figure 5.8: Number of frames vs. mean average precision, only using CLIP features.

In summary, the success of our genre classification models is due to combining diverse pretrained features, leveraging all video scenes and the full audio track, and avoiding temporal averaging. Instead, we feed the complete feature sequences into a transformer, which excels at handling long sequences. These design choices are feasible thanks to the large scale of the MovieNet dataset.

Our work on cinematic genre classification highlights the potential for enhancing video emotion classification, especially as large-scale video datasets with emotion labels become available in the future. For now, due to the limited size of existing video emotion datasets, the emotion classifier in our video-based music generation system uses a fixed, smaller set of input frames.

We are now ready to bring all components together to realize our video-based music generator. This system combines modules for video emotion analysis and temporal analysis (via scene cut detection) with a music generator that incorporates both emotion and temporal conditioning. The final MIDI output is synthesized into audio using the Fluidsynth software. (As a reminder, Figure 1.1, introduced in the Introduction, illustrates the full system architecture.) We detail the integration process in the following chapter.

Chapter 6

Video-based music generation

In this chapter, we explain how we integrate all the components to create our video-based music generator. Specifically, we combine the four modules of our two-stage, two-branch pipeline: the temporal boundary conditioner for music generation (Section 4.1), the emotion conditioner for music generation (Section 4.2), the video scenecut (boundary) extractor (Section 5.1), and the the video emotion classifier (Section 5.2.2). We name our model *EMSYNC*, as it aligns music with video by matching their **em**otions and **sync**hronizing their temporal boundaries.

Before we proceed, we address the issue of incompatible emotion representations between the video emotion classifier (VEMOCLAP) and emotion-based music conditioning. While VEMOCLAP outputs probability distributions for discrete emotion categories, the music generator is conditioned on emotions represented as continuous valence and arousal values, ranging from -1 to 1.

6.1 Emotion mapping

We establish a correspondence between the emotion representations in our video analysis and music generator models. Specifically, we map discrete emotions to continuous valence-arousal values.

Russell and Mehrabian (1977) conducted user studies to determine the corresponding valence and arousal values for discrete emotion categories. They presented their findings in a table containing the mean and standard deviation values of valence and arousal for each categorical emotion. We extract and use the values corresponding to Ekman's six basic emotions, as shown in Table 6.1 (Ekman, 1971).

Using the output probabilities of the video classifier along with the means and standard deviations for each emotion, we construct a mixture of Gaussian distributions. Users can either sample from the mixture—by first selecting an emotion category based on the output probabilities and then sampling from the corresponding Gaussian distribution—or use the mean of the mixture, which is calculated as the weighted average of the emotion category means, with weights determined by the output probabilities.

Table 6.1: Means and standard deviations of valence and arousal values corresponding to categorical emotions of Ekman (1971), obtained from the user studies of Russell and Mehrabian (1977).

	Valence		Arousal	
Emotion	Mean	Std	Mean	Std
anger	-0.51	0.20	0.59	0.29
disgust	-0.60	0.20	0.35	0.41
fear	-0.64	0.20	0.60	0.32
joy	0.76	0.22	0.48	0.26
sadness	-0.63	0.23	-0.27	0.34
surprise	0.40	0.30	0.67	0.27

Additionally, we introduce a parameter that represents the maximum absolute value of means across all emotion categories. Using this parameter, we compute shifting and scaling parameters that are applied to all valence-arousal distributions. This approach enables users to adjust the conditioning of the music generator, choosing between a wider or narrower range of emotions. The following equations show how we readjust the ranges of valence and arousal values.

$$\mathbf{v}'_{\text{means}} = \frac{2r \cdot (\mathbf{v}_{\text{means}} - \min(\mathbf{v}_{\text{means}}))}{\max(\mathbf{v}_{\text{means}}) - \min(\mathbf{v}_{\text{means}})} - r$$
$$\mathbf{v}'_{\text{stds}} = \frac{2r \cdot \mathbf{v}_{\text{stds}}}{\max(\mathbf{v}_{\text{means}}) - \min(\mathbf{v}_{\text{means}})}$$
$$\mathbf{a}'_{\text{means}} = \frac{2r \cdot (\mathbf{a}_{\text{means}} - \min(\mathbf{a}_{\text{means}}))}{\max(\mathbf{a}_{\text{means}}) - \min(\mathbf{a}_{\text{means}})} - r$$
$$\mathbf{a}'_{\text{stds}} = \frac{2r \cdot \mathbf{a}_{\text{stds}}}{\max(\mathbf{a}_{\text{means}}) - \min(\mathbf{a}_{\text{means}})}$$

We select r to represent the new maximum absolute value of means across all emotion categories, and hence, the overall range becomes $2r$. The prime superscript signify new values. Bold letters represent the vectors that contains the mean and standard deviation values of valence and arousal of all emotion categories. While we shift and scale the means, we only scale the standard deviations as standard deviation only represents the spread of the data and not the positioning. The minimum and the maximum values can be found directly from Table 6.1 as:

$$\min(\mathbf{v}_{\text{means}}) = -0.64, \quad \max(\mathbf{v}_{\text{means}}) = 0.76, \quad \min(\mathbf{a}_{\text{means}}) = -0.27, \quad \max(\mathbf{a}_{\text{means}}) = 0.67.$$

6.2 Implementation details

The video-based music generation pipeline is shown in Figure 6.1. In the left branch, we perform emotion-conditioning. We begin by setting the maximum absolute value of the means across all

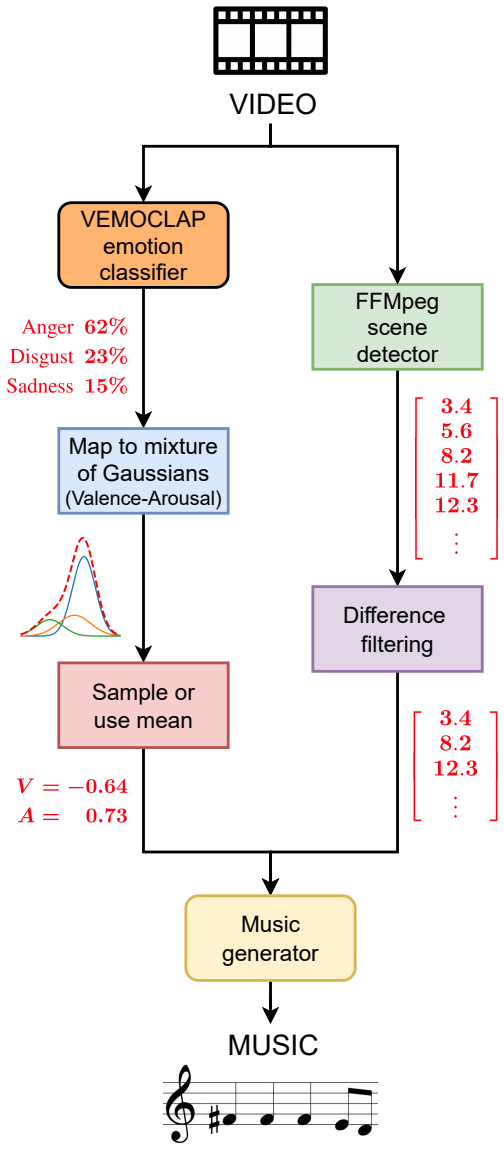


Figure 6.1: Our video-based music generation pipeline. The values next to the arrows are sample values.

emotion categories to 0.8 and adjust the valence-arousal distributions accordingly. The VEMOCLAP model classifies the video's emotions and outputs a probability distribution. This distribution is then mapped to a mixture of Gaussians, with the probability distribution serving as the weights for the valence-arousal distributions of each emotion category. From this, we can either sample from the distribution or use the mean of the mixture. To ensure more robust results, we choose to use the mean of the mixture. The final output consists of two values for valence and arousal, which become the emotion conditions for the music generator.

The right branch focuses on temporal conditioning. Here, the FFMpeg scene detector identifies scene cut positions by detecting when the average pixel difference between consecutive frames

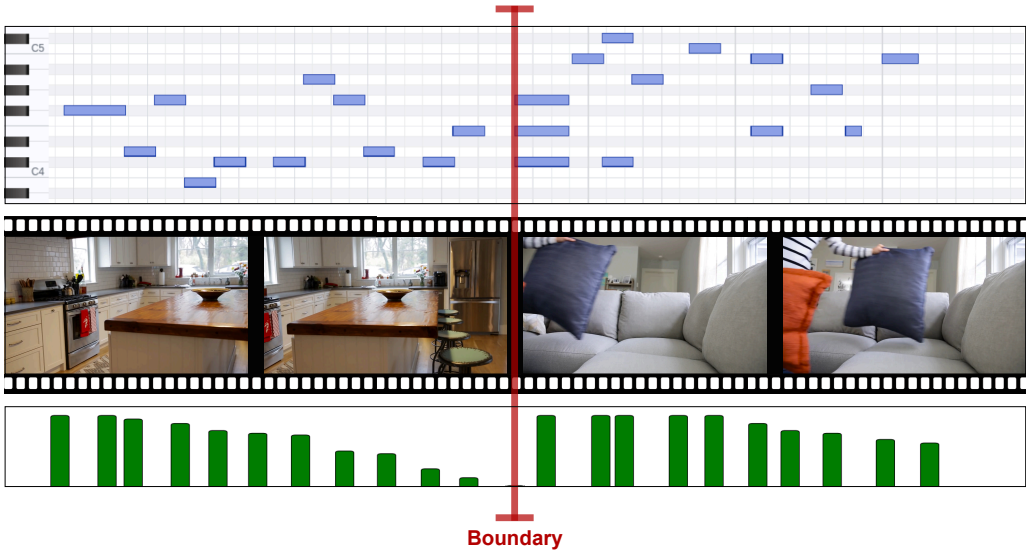


Figure 6.2: Graphical illustration of boundaries.

exceeds 27% of the pixel value range (0-255). A difference filter is then applied to remove positions that are too close together, preventing a cacophony of overlapping strong chords placed too near each other. In our experiments, we discard scenes that are less than 4 seconds apart. The resulting timestamps serve as the temporal conditions for the music generator.

The music generator produces tokens until the generated music matches the video's duration. After the MIDI is synthesized into an audio waveform, a 3-second fade-out is applied to avoid an abrupt ending. After training the video emotion classifier and the music generator—along with its temporal and emotional conditioning mechanisms—we do not perform any additional training or fine-tuning on the combined model. All hyperparameters are determined through trial and error, as well as manual inspection of the outputs.

Figure 6.2 provides a graphical illustration of how we align musical boundaries (strong chords) with video boundaries (scene cuts). During the training of our boundary conditioning mechanism, we use the timestamps of long-duration chords as target boundaries. In video-based inference, these target boundaries are replaced with the timestamps of scene cuts extracted from the video.

In the top row, we present a piece of symbolic music, where a chord consisting of three or more simultaneous notes with a duration exceeding a set threshold of two beats defines a musical boundary. These musical boundaries are only used during training, as explained in Section 4.1. The middle row shows a video, where scene cuts serve as video boundaries and are used during inference. The bottom row illustrates the boundary offsets, which are calculated based on the input boundaries.

It is important to note that these figures are for illustrative purposes, and in this example, the offsets do not perfectly align with the music, except at the boundaries. Additionally, during video-based inference, we do not expect the music and video boundaries to match perfectly. As previously mentioned, our music generator creates strong chords in the vicinity of the input boundaries, ensuring rhythmic coherence with the rest of the generated music without disrupting

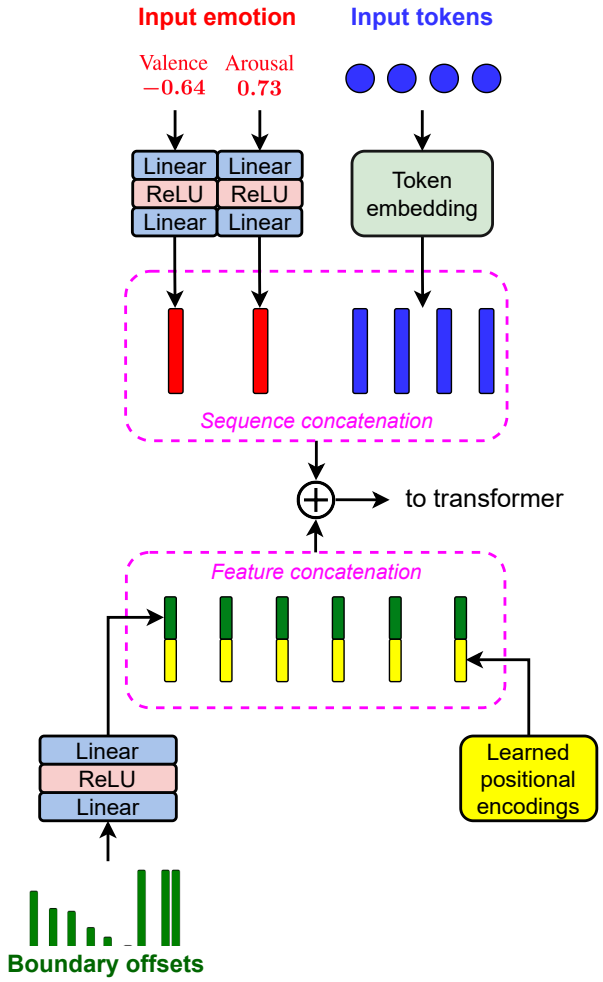


Figure 6.3: Our music generator. Numbers underneath valence and arousal are sample values.

its flow.

For emotion-based MIDI generation, we use our continuous-token model described in Section 4.2. We select this model because it utilizes continuous-valued conditions, avoiding the loss of information associated with discretization. Additionally, while its performance is nearly on par with our top-performing continuous-concatenated model, it does not increase the model's dimensionality. However, all of our emotion conditioning modules are compatible for integration into the overall model.

Figure 6.3 illustrates the overall conditional music generation pipeline, integrating both emotional and temporal conditioning. Emotion conditioning is achieved by concatenating conditioning vectors with token embeddings along the sequence dimension. As outlined in Section 4.1, temporal boundary conditioning is implemented by concatenating boundary offsets with positional encodings along the feature dimension. The transformer's input is represented as:

$$\mathbf{X}_{\text{trans}} = \text{Concat}_s\left(\text{FFN}_v(x_v), \text{FFN}_a(x_a), \text{Embedding}(\mathbf{x}_t)\right) + \text{Concat}_f\left(\text{FFN}_b(\mathbf{b}), \mathbf{W}_{\mathbf{pe}}\right) \tag{6.1}$$

where x_v and x_a are the valence and arousal inputs, $\mathbf{x}_t$ is the input token sequence, $\mathbf{b}$ represents the input boundary offsets, and $\mathbf{W_{pe}}$ is the learned positional encoding. The feed-forward networks FFN_v, FFN_a, and FFN_b process the valence, arousal, and boundary offset inputs, respectively. The operations Concat_s and Concat_f denote concatenation along the sequence and feature dimensions, respectively. These two conditioning mechanisms are designed to be easily integrated without interfering with each other. Furthermore, neither of the conditioning mechanisms modifies the transformer's body, making it easy to fine-tune our trained models for other types of music generation tasks by simply removing or replacing the conditioning modules.

We choose the size of our music generator to ensure a fair comparison against the state-of-the-art methods which we detail later. Our music generator consists of 11 layers, 8 attention heads, and a dimensionality of 512, totaling 37M parameters. We train it with a context length of 1216 tokens and a batch size of 64. Training is performed using cross-entropy loss with a learning rate of 2e-4 for the first 300k steps, followed by 5e-5 for the next 300k steps. We use the Adam optimizer with gradient clipping to a norm of 1 (Kingma & Ba, 2015). We do not apply any regularization methods and do not observe overfitting, as the model is trained on large-scale, densely labeled data where it predicts each token of its input sequence.

6.3 Evaluation

As discussed in Section 2.6, conducting an objective evaluation is challenging when using unmatched video and MIDI datasets. We therefore focus on the subjective evaluation of our overall model. We conduct a user study to compare our results with two open-source state-of-the-art methods that generate MIDI from arbitrary videos: Video2Music (Kang et al., 2024) and CMT (Di et al., 2021), both previously introduced in Section 2.6.

Video2Music shares similarities with our approach in leveraging both low-level video features and high-level emotional conditioning. However, a major difference lies in the music generation process: their model generates only chord sequences, which are then transformed into piano melodies using fixed, hand-crafted arpeggiation patterns. These patterns all rely on a coarse time grid, limiting their ability to capture the expressive timing of human performance. In contrast, our model directly generates multi-instrument music in a direct and learned manner, using an event-based representation with fine temporal precision. Moreover, Video2Music requires the user to provide the musical key and initial chords, while our method is fully automatic.

The Controllable Music Transformer (CMT) generates multi-instrument music directly and is trained on the Lakh Pianoroll Dataset, similar to our approach. However, unlike our model, it relies solely on low-level video features such as timing, motion saliency, and motion speed, without incorporating high-level features like emotion. Additionally, similar to Video2Music, it restricts note placement to beat subdivisions, limiting its ability to capture expressive timing nuances.

For our final evaluation, where we compare our results to Video2Music and CMT, we set our model size to be comparable to theirs: our model has 37M parameters, compared to 39M in CMT

and 33M in Video2Music. For all methods, we synthesize output MIDI files into audio waveforms using Fluidsynth and apply peak normalization up to -3 dB.

6.3.1 Dataset

We train our music generator on the Lakh Pianoroll-5 dataset (Dong et al., 2018). For evaluation, we perform inference on videos from the Pittsburgh Advertisements Dataset (Ads) (Hussain et al., 2017). We select this dataset because advertisements often rely on music to maximize viewer engagement (Zander, 2006). Additionally, since none of the compared methods have used this dataset, it ensures a fair comparison. Finally, the Ads dataset includes sentiment labels, allowing us to construct a well-balanced test set in terms of emotion.

Our emotion classifier uses the six emotion categories proposed by Ekman (1971). To avoid bias in favor of our method, we construct the evaluation set using a different emotion categorization. We start by filtering the Ads dataset to select videos associated with the four basic emotions commonly used in music emotion classification: cheerful, calm, angry, and sad (Juslin & Sloboda, 2001). These four emotions also correspond to the quadrants of the valence-arousal plane (Russell, 1980), ensuring comprehensive emotional coverage. We then exclude videos shorter than one minute. To ensure unbiased selection, we use randomly generated YouTube IDs. These IDs are sorted alphabetically, and we select the first six videos from each emotion category, resulting in a total of 24 test videos. Finally, all videos are trimmed to a uniform duration of one minute.

6.3.2 Survey

Using the 24 test videos, we generate accompanying music with the three methods and create 12 survey pages, each containing two videos. For each video, the three music versions, generated by the compared methods, are presented side by side with anonymized method names, with each methods's output appearing equally in the left, center, and right positions. We enroll 36 participants—17 with self-reported knowledge of music theory (Group 1) and 19 without (Group 2). Each survey page and video is viewed by three participants, with at least one participant from each group represented.

A screenshot of part of our survey is shown in Figure 6.4. The model names are anonymized as Method A, Method B, and Method C. Users are asked to rank the methods by dragging the method names from the left-hand side to the right-hand side, ensuring that missing a question does not result in an error. For readers, we also provide output samples from eight videos online, with method names deanonymized and each method's output appearing in the same position for clarity.[1]

We ask participants to rank the three methods based on the standard criteria used in previous works (Di et al., 2021; Kang et al., 2024): *Music richness (MR)*: The richness and diversity of the music, independent of the video content. *Music quality (MQ)*: The overall quality of the music, independent of the video content. *Emotion match (EM)*: How well the music matches the video in

[1] https://em-sync.github.io/

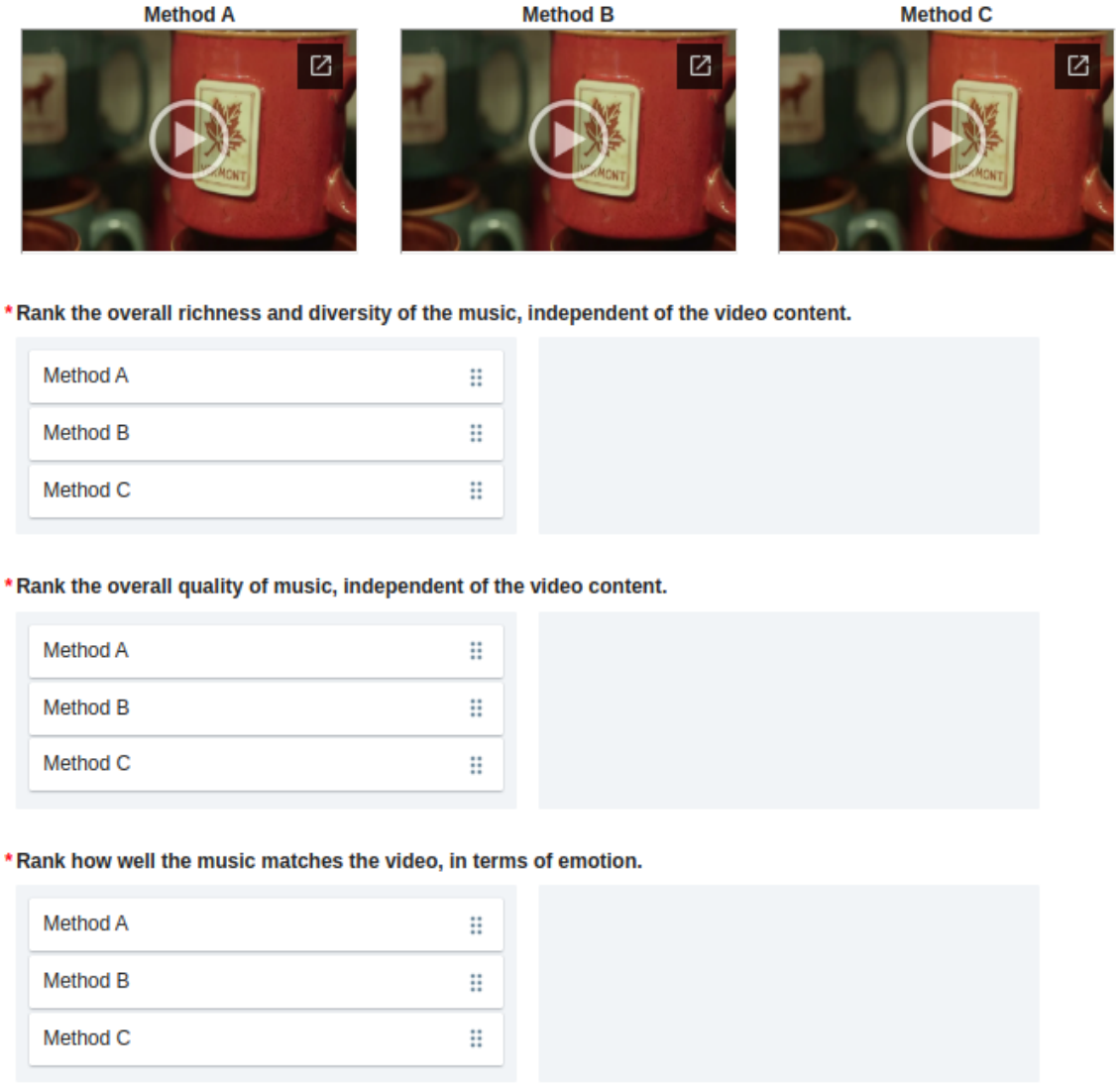


Figure 6.4: A screenshot of part of our video-based music generation survey.

terms of emotion. *Timing match (TM)*: How well the music synchronizes with the video in terms of rhythm and timing. *Overall match (OM)*: How well the music matches the video as a whole. Each model is assigned a unique ranking of 1 (best), 2, or 3 (worst), ensuring no ties. Finally, we report the runtimes as the average time required to generate music for a one-minute video, including model initialization.

6.4 Results and discussion

We compare CMT (Di et al., 2021), Video2Music (Kang et al., 2024), and our method EMSYNC across all subjective criteria using average rankings. Table 6.2, Table 6.3, and Table 6.4 present the results for Group 1, Group 2, and all participants, respectively. Rankings are first averaged per video and then averaged across all videos. A lower score indicates a better ranking (1 being best, 3 worst). Standard deviations across videos are shown in parentheses.

Table 6.2: Results for participants with a self-reported understanding of music theory. Lower values indicate better rankings (1 = best, 3 = worst). Values are reported as mean (standard deviation).

	Music Richness	Music Quality	Emotion Match	Timing Match	Overall Match
CMT (Di et al., 2021)	1.85 (0.56)	2.44 (0.60)	2.00 (0.53)	2.12 (0.76)	2.08 (0.64)
Video2Music (Kang et al., 2024)	2.52 (0.71)	1.85 (0.67)	2.12 (0.86)	1.96 (0.76)	2.17 (0.83)
EMSYNC (Ours)	**1.62** (0.71)	**1.71** (0.76)	**1.88** (0.85)	**1.92** (0.75)	**1.75** (0.79)

Table 6.3: Results for participants without a self-reported understanding of music theory. Lower values indicate better rankings (1 = best, 3 = worst). Values are reported as mean (standard deviation).

	Music Richness	Music Quality	Emotion Match	Timing Match	Overall Match
CMT (Di et al., 2021)	1.79 (0.64)	2.56 (0.61)	2.46 (0.57)	2.31 (0.75)	2.35 (0.58)
Video2Music (Kang et al., 2024)	2.71 (0.36)	1.77 (0.53)	1.85 (0.73)	1.98 (0.65)	2.08 (0.73)
EMSYNC (Ours)	**1.50** (0.49)	**1.67** (0.67)	**1.69** (0.64)	**1.71** (0.69)	**1.56** (0.66)

Figures 6.5, 6.6, and 6.7 present stacked bar charts showing the frequencies of participant responses from Group 1, Group 2, and all participants, respectively. Each chart displays the frequency of each rank assigned, grouped by evaluation metric and method.

Finally, we report the average runtime for each model when generating music for a one-minute video, including model initialization: EMSYNC takes 1.42 minutes, CMT 3.55 minutes, and Video2Music 1.61 minutes.

Table 6.4: Results for all participants. Lower values indicate better rankings (1 = best, 3 = worst). Values are reported as mean (standard deviation).

	Music Richness	Music Quality	Emotion Match	Timing Match	Overall Match
CMT (Di et al., 2021)	1.82 (0.60)	2.50 (0.60)	2.23 (0.59)	2.22 (0.75)	2.22 (0.62)
Video2Music (Kang et al., 2024)	2.61 (0.57)	1.81 (0.60)	1.99 (0.80)	1.97 (0.70)	2.12 (0.78)
EMSYNC (Ours)	**1.56** (0.61)	**1.69** (0.71)	**1.78** (0.75)	**1.81** (0.72)	**1.66** (0.73)

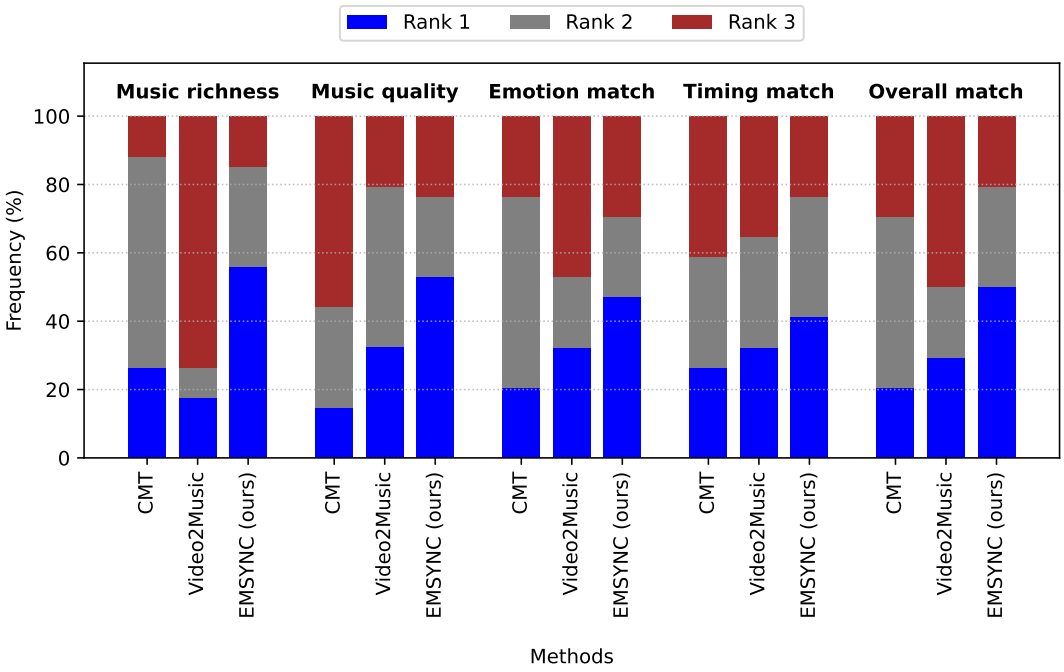


Figure 6.5: Stacked bar chart results for participants with a self-reported understanding of music theory.

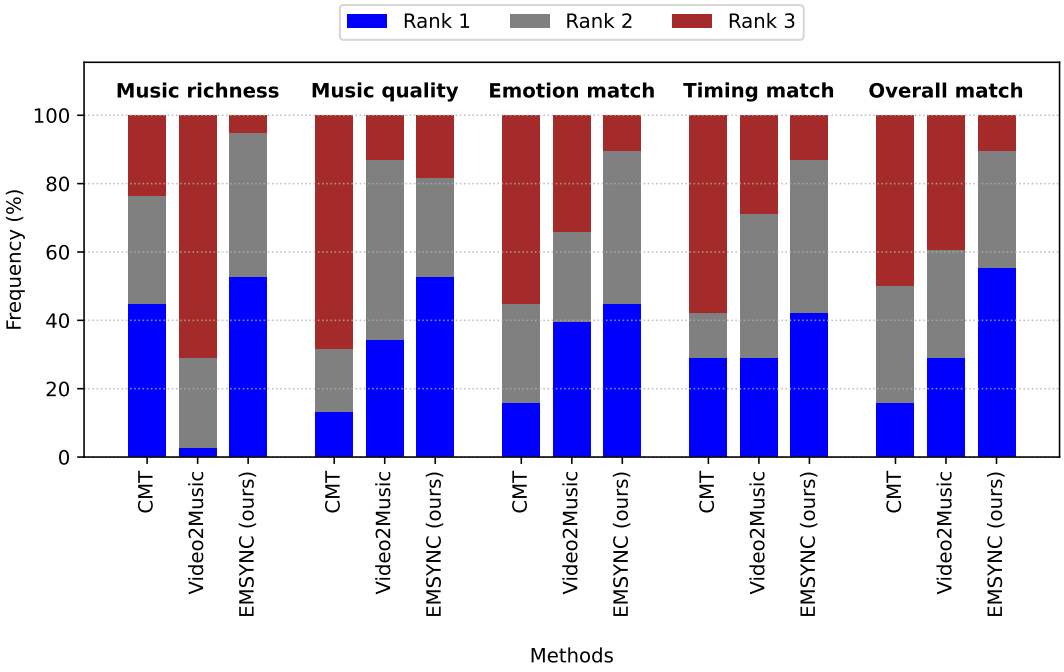


Figure 6.6: Stacked bar chart results for participants without a self-reported understanding of music theory.

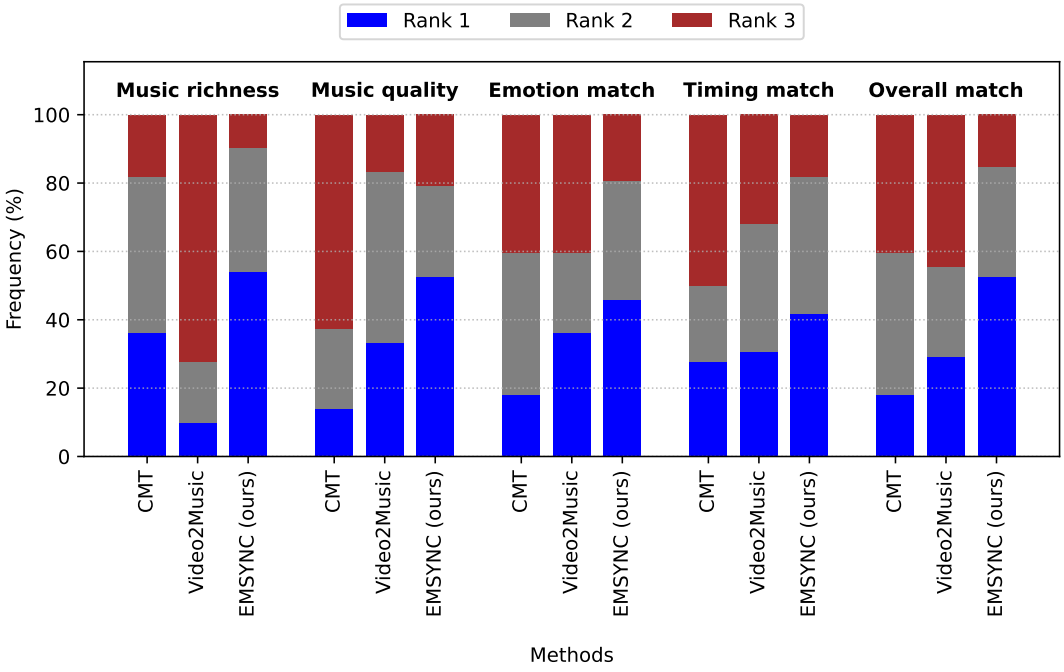


Figure 6.7: Stacked bar chart results for all participants.

In terms of music richness, Video2Music receives the worst average ranking across both participant groups. This may be attributed to its reliance on fixed arpeggiation patterns or its limitation to piano-only output. Participants with music theory knowledge assign the best rank to CMT less frequently. A possible explanation, based on our preliminary review of model outputs, is that CMT adjusts note density using sparse constraints such as video motion, which can lead to rhythmically incoherent results. Participants without music theory knowledge may perceive this as adding variety and contributing to musical richness, while others may not. Additionally, participants without music theory knowledge seldom rank Video2Music first, possibly because it is readily apparent, even without any music theory knowledge, that the output lacks variety due to its restriction to a single instrument, whereas the other methods produce multi-instrumental music.

In terms of music quality, CMT receives the worst average ranking from both participant groups. This may again be due to its continual modulation of note density. Regarding emotion match, in both groups, CMT is the least frequently assigned the best rank, which is expected given that it relies solely on low-level features and lacks high-level conditioning such as emotion. However, a surprising observation is that participants with music theory knowledge are also the least likely to assign CMT the worst rank. As a result, its average ranking in this category is better than that of Video2Music. This may highlight the inherent difficulty of high-level video emotion classification and suggests that there is still room for improvement in the emotion analysis components of both our model and Video2Music.

In terms of timing match, our method is particularly favored by participants without music theory knowledge. This may support our hypothesis that alignment with sparse temporal cues, such as scene cuts, is more easily perceived by the general public. In terms of overall match, the effectiveness of our method becomes even more evident. It achieves the best performance across all participant groups, while the average rankings for the other methods remain worse than 2.

EMSYNC consistently outperforms other methods across all metrics for all groups. These results confirm that EMSYNC produces the most diverse and high-quality music with the best emotional alignment, temporal synchronization, and overall video compatibility while also being the most computationally efficient model.

This chapter concludes the presentation of our video-based music generation system in its entirety. Readers may recall our initial overview of the model pipeline in Figure 1.1, and note that we have now discussed each component in detail—except for the audio generator. To build a working prototype capable of producing listenable outputs, we used preset sound libraries together with the Fluidsynth software, which maps these sounds to individual MIDI notes. The main limitation of this approach is the lack of timbral diversity: because the sound library is fixed, the resulting audio lacks nuance and realism, often sounding overly mechanical compared to performances by human musicians.

While several existing methods generate music directly in the audio domain, they are typically conditioned only on text and operate as black-box models (Copet et al., 2023; Evans et al., 2024; Lam et al., 2023; H. Liu et al., 2024). Since their outputs exist purely as audio, they are not editable or interpretable. As such, it may be more accurate to describe their outputs as musical sounds

rather than musical compositions. This black-box approach contrasts with the human process of music creation, which typically involves two distinct stages: composition (defining chords, notes, melodies, and structure) followed by production (generating audio through performance). We argue that a promising direction for automatic music generation is to employ DNNs separately for both stages—composition via MIDI generation, and production via MIDI-conditioned audio generation. Although research into MIDI-based audio generation is ongoing, current efforts are often limited to single-track piano music, largely due to the scarcity of paired multi-instrument MIDI and audio datasets (Hawthorne et al., 2019; Sambaragi et al., 2024). In the next chapter, we present a preliminary analysis of audio generation and its challenges, particularly when relying on synthetic data to compensate for the lack of real-world audio recordings.

Chapter 7

Audio bandwidth extension

As our video-based music generator creates soundtracks in the symbolic music format, they are synthesized into audio using computer software and a soundfont library, which includes fixed sounds for particular pitches of particular instruments. However, the end result is very mechanical and unrealistic, compared to the music produced by human musicians. We therefore investigate audio enhancement to take the first step towards improving the quality of audio synthesized from symbolic music.

There are currently no multi-instrument music datasets with paired audio (performed by humans) and MIDI, which would be required to train an end-to-end model capable of generating audio that sounds like human performance from synthesized MIDI. Even if such a dataset were available, creating human-like audio from MIDI would still pose an ill-defined problem, as an infinite number of human performances can be derived from the same music composition (i.e., MIDI file). In the absence of a multi-instrumental dataset with MIDI-audio pairs, one potential approach is to train DNNs using synthetic audio generated from MIDI via sound libraries. However, the quality of the output from these trained models would be limited by the quality of the synthesized audio, rendering the DNN useless.

We investigate another ill-defined audio enhancement problem, namely audio bandwidth extension. This task deals with creating full-band audio from band-limited audio. The information from the limited bands are missing, and there can be multiple candidates for this missing information, making this task ill-defined. We also explore this problem due to the use of synthetic data. Overall, studying audio bandwidth extension provides valuable insights that are applicable to other ill-defined audio generation tasks that utilize synthetic data, such as MIDI-to-audio generation. A key advantage of investigating audio bandwidth extension is the abundance of data available, as one can generate an unlimited amount of input data by processing any full-band audio with various low-pass filters.

When considering audio bandwidth extension for the enhancement of real-world recordings, no full-bandwidth version exists and as such there is no "ground-truth" to act as a target for DNNs. To this end, training data is typically obtained by low-pass filtering full-bandwidth recordings. However, since real-world band-limited samples are not the result of some hypothetical universal

digital low-pass filter, it can be challenging to develop robust techniques for bandwidth extension which rely on a loose approximation of the bandwidth reduction process, and in turn to generalize to unseen recordings. While the trained DNNs can perform well on training data created with one type of low-pass filter, they may fail to generalize to audio content subjected to different types of low-pass filter. Throughout our work, we label this as *filter overfitting* which can be understood as a lack of *filter generalization*.

While the use of low-pass filtering is widespread among existing work on audio bandwidth extension using DNNs, to the best of our knowledge, no work to date as thoroughly investigated the topic of filter generalization. We argue that the lack of generalization to various types of signal deterioration is an important challenge in creating audio enhancement models for real-world deployment. In our work, we present a rigorous analysis of filter generalization, evaluating generalization to different filters used to preprocess input data, on the task of bandwidth extension of complex music signals, using two popular DNN architectures.

To evaluate sample overfitting, we use a testing data split different from the training data split, which is *unseen data* for the trained models. Additionally, to evaluate filter generalization, we preprocess the testing input data, with a filter that does not match the filters that preprocess the training input data, i.e., an *unseen filter* and compare it to the test setting where the filters preprocessing the training and testing data do match, i.e, *seen filters*. We argue that testing with the unseen filter can be thought of as a representation of real-world signal degradation, in which the true underlying degradation function is unknown.

We evaluate two different regularization methods that are used in the literature to increase generalization. In particular, we compare the usage of data augmentation, batch normalization, and dropout, against the baseline of not using any regularization methods. We introduce a novel data augmentation technique of using a set of different low-pass filters to preprocess the input data, in which the unseen test filter is never present. We examine the training process by tracking the model's performance throughout training iterations, by performing validation using both seen and unseen filters.

One of the DNN models we employ is the *U-Net*, which was first used for biomedical image segmentation (Ronneberger et al., 2015), and later in audio signal processing tasks such singing voice separation (Jansson et al., 2017), and eventually for audio enhancement (S. Kim & Sathe, 2019; Kuleshov et al., 2017; T.-Y. Lim et al., 2018; Macartney & Weyde, 2018). In addition to the U-Net, we also use the deep residual network model (*ResNet*) (He et al., 2016) since it is one of the most widely used DNN architectures in signal processing tasks. Even though the U-Net is a popular architecture in the recent audio processing literature, to the best of our knowledge, no work in the domain of audio processing compares the U-Net against the well-established baseline of the ResNet. A small number of comparative studies exist in the fields of image processing and medical imaging, in which either the number of parameters of the compared models is not stated (Rempfler et al., 2017), or in which the ResNet has significantly fewer parameters than the U-Net (Chiou et al., 2018; Ghodrati et al., 2019; S. Wu et al., 2018). In all these works, the ResNet outperforms the U-Net by a small margin. We also present a comparison between the U-Net and

ResNet, where each has a similar number of parameters.

Our main findings indicate that filter overfitting occurs for both the U-Net and ResNet, although to different degrees, and that the use of multi-filter data augmentation, as opposed to more traditional regularization techniques, is a promising means to mitigate this overfitting problem and thus improve filter generalization for bandwidth extension.

We continue by presenting our methodology in detail, starting with the models we employ.

7.1 Models

In this section, we define the two baseline models: U-Net and ResNet. For both models, we follow the approach of Kuleshov et al. (2017) and use raw audio as the input rather than time-frequency transforms (e.g., as in Miron and Davies (2018)). As such we remove any need for phase reconstruction in the output. However, since we address bandwidth extension and not audio super-resolution per se, our inputs are not subsampled. Hence the sizes of the input and the output are equal for all our models.

7.1.1 U-Net

The U-Net architecture (Ronneberger et al., 2015), like the its predecessor the auto-encoder, consists of two main groups. The first group contains downsampling layers and followed in the second group by upsampling layers, as shown in Figure 7.1 (left). In the U-Net, individual downsampling and upsampling layers at the same scale are connected through stacking connections, e.g., the output of the first downsampling convolutional block is stacked with the input of the last upsampling convolutional block. As is common in enhancement models, an additive connection from the input to the output is also used, so that the network only needs to model the *difference* between the input and the target signals, rather than creating the target signal from scratch.

In the downsampling group, one-dimensional convolutional layers with stride 2 are used, effectively halving the activation length. Borrowing from image processing terminology, the upsampling group includes "sub-pixel" layers (also known as the pixel shuffler) (Shi et al., 2016) to double the activation length. Sub-pixel layers work by weaving the samples in the spatial dimension, taken from alternate channels, effectively halving the channel length. The number of parameters is selected to replicate the original work using U-Net for audio super-resolution (Kuleshov, 2020; Kuleshov et al., 2017), which we denote as *Audio-SR-U-Net* throughout. This resulted in a network with 56.4 million parameters.

7.1.2 ResNet

A common issue with training vanilla feed-forward neural networks with many layers is the "vanishing gradient" problem, in which the gradient back-propagated to the earliest layers approaches zero, due to repeated multiplications. Residual networks (He et al., 2016) eliminated this problem by using *residual blocks*, which only model a fraction of the difference between their inputs

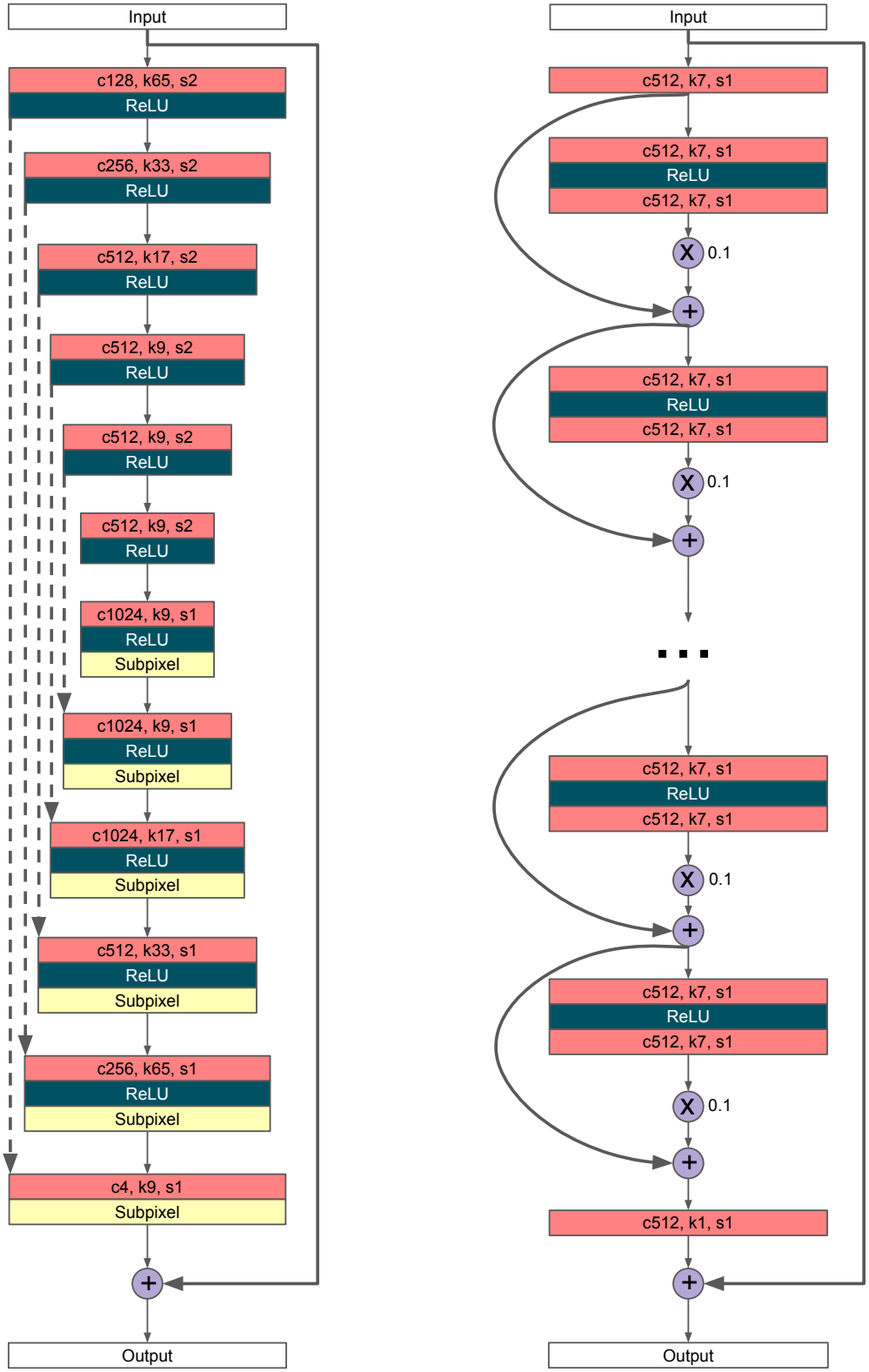


Figure 7.1: Audio bandwidth extension models used. (Left) U-Net model, where dashed lines indicate stacking connections. (Right) ResNet model with 15 residual blocks. c, k, and s indicate channel size, kernel size, and stride of the convolutional layers, respectively.

and outputs. Commonly, each residual block includes two convolutional layers and a nonlinear function in between them. Very deep models include *residual scaling* in which the output of each residual block is multiplied by a small number, e.g., 0.1, and then summed with its input, to further stabilize training. Our ResNet model is represented in Figure 7.1 (right).

Unlike the U-Net, the ResNet activation lengths stay constant throughout the network. In this way we can avoid any loss of temporal information since our goal is to create a high-resolution output of equal length to the input. Note that we use a simple design where all convolutional layers except the last one have the same number of parameters. Similar to the U-Net implementation, it has 55.1 million parameters.

In all our models, all convolutions apply the appropriate zero padding to keep the activation sizes constant. This is even true for downsampling convolutions since the downsampling effect is achieved using strided convolutions.

To analyze generalization, we present ablation studies, in which we incorporate common methods to avoid overfitting, defined as *regularization methods*.

7.2 Regularization methods

We now detail the standard regularization methods such as dropout and batch normalization, as well as our novel data augmentation strategy.

7.2.1 Dropout

One of the simplest and most common methods to prevent overfitting is dropout, where activation units are dropped based on a fixed probability (Srivastava et al., 2014). This introduces noise within the hidden layers and prevents the neurons from excessive co-adaptation.

Even though dropout has been largely superceded in the literature by batch normalization, especially in residual networks, new state-of-the-art residual models, namely wide residual networks (Zagoruyko & Komodakis, 2016) do employ it. Furthermore, *Audio-SR-U-Net*'s open-source implementation[1] uses a dropout layer instead of batch normalization and thus, we followed this implementation in our U-Net model and used dropout layers after each upsampling convolutional layer. In our ResNet model, we placed dropout layers between the two convolutional layers of each residual block. For all experiments, we set the dropout rate at 0.5.

7.2.2 Batch normalization

While training DNNs, updating the parameters of the model effectively changes the distribution of the inputs for the next layers. This is defined as *internal covariate shift* and batch normalization addresses this problem by normalizing the layer inputs (Ioffe & Szegedy, 2015). Even though batch normalization is mainly proposed to speed up training, it provides regularization as well. Because the parameters for the normalization are learned based on each batch, they can only

[1] https://github.com/kuleshov/audio-super-res

provide a noisy estimate of the true mean and variance. Normalization using these estimated parameters introduces noise within the hidden layers and reduces overfitting.

For the U-Net, we followed *Audio-SR-U-Net* model (Kuleshov et al., 2017) and inserted batch normalization layers after each downsampling convolutional layer. For the ResNet, batch normalization is used after each convolutional layer.

7.2.3 Data augmentation

To increase sample generalization of DNNs, data augmentation is used, where the input data samples are transformed before being fed into the DNN, effectively increasing the number and diversity of training samples. Data augmentation is very common in image-based tasks, and mostly utilizes geometric transformations such as rotating, flipping, or cropping (Taylor & Nitschke, 2018). Geometric transformations of this kind when applied to music signals typically do not produce realistic samples and while some work has been conducted on data augmentation for musical signals (McFee et al., 2015) it primarily targets robustness for classification tasks such as instrument identification via lossy transformations including time-stretching and pitch shifting, which cannot be applied in this context.

Since our main goal is to explore and then improve filter generalization, we propose a data augmentation method where many different types of filters are used during training. Our baseline methods without data augmentation, uses a *single-filter* training setting, specifically a 6th order Chebyshev Type 1, denoted "Chebyshev-1, 6". Whereas using data augmentation, in a *multi-filter* training setting, we adopt a set of eight different filters, picked randomly for each input sample during training. These eight filters consist of *Chebyshev*-1, *Bessel*, and *Elliptic* filters of different orders. To evaluate filter generalization, we reserve the 6th order *Butterworth* filter as the unseen filter. The filters are summarized in Table 7.1, and their usage during evaluation is detailed in Section 7.4. A graphical overview of their different frequency magnitude responses is shown in Figure 7.2.

Combining the three regularization methods with the two baseline architectures we obtain the following eight models which are used in our experiments: baseline U-Net (*U-Net*), U-Net with data augmentation (*U-Net DA*), U-Net with batch normalization (*U-Net BN*), U-Net with dropout (*U-Net DO*), baseline ResNet (*ResNet*), ResNet with data augmentation (*ResNet DA*), ResNet with batch normalization (*ResNet BN*), and ResNet with dropout (*ResNet DO*).

7.3 Dataset

Machine learning approaches to bandwidth extension formulate the problem via the use of datasets that contain both full-bandwidth (high-quality) and band-limited (low-quality) versions of each audio signal. A straightforward way to construct these pairs of samples is to obtain a high-quality dataset and then to low-pass filter it. Even though there are many musical audio datasets, especially within the music information retrieval community, many of them are collated from diverse

Table 7.1: The types and orders of the low-pass filters used, under two different training settings, *single-filter* (no data augmentation) and *multi-filter* (data augmentation).

	Single-filter (No data augmentation)	Multi-filter (Data augmentation)
Training	Chebyshev-1, 6	Chebyshev-1, 6 Chebyshev-1, 8 Chebyshev-1, 10 Chebyshev-1, 12 Bessel, 6 Bessel, 12 Elliptic, 6 Elliptic, 12
Validation with seen filter(s)		
Validation with unseen filter	Butterworth, 6	Butterworth, 6
Testing with unseen filter		
Testing with seen filter	Chebyshev-1, 6	Chebyshev-1, 6

sources (including researchers' personal audio collections) and often contain audio content has been compressed (e.g. via lossy MP3/AAC encoding), hence they are not strictly full-bandwidth.

Other than the need for full-bandwidth musical audio content, our proposed approach is intended to be agnostic to musical style. To this end, any uncompressed full-bandwidth musical content could be used as training material, however in order to allow reproducibility, we select the following two publicly available datasets, which contain full-bandwidth, stereo and multi-track musical audio: MedleyDB (version 2.0) (Bittner et al., 2016) and DSD100 (Liutkus et al., 2017). In each dataset, the audio content is sampled at 44100 Hz (i.e., CD quality), and hence its bandwidth is 22050 Hz.

MedleyDB consists of 121 songs, while DSD100 has two splits for training and testing, each containing 50 songs. Given the inclusion of isolated multi-track stems, both datasets have found high uptake in music mixing and audio source separation research. However, we seek to address bandwidth extension for multi-instrument music as opposed to isolated single instruments, and thus we retain only the stereo mixes of each song. To create band-limited input samples, we apply a low-pass filter with a fixed cut-off of 11025 Hz, i.e., half the bandwidth of the original.

The DSD100 test split is used for testing, the last 8 songs of DSD100 training split are used for validation, with all remaining songs of DSD100 training split plus the entire MedleyDB dataset used for training.

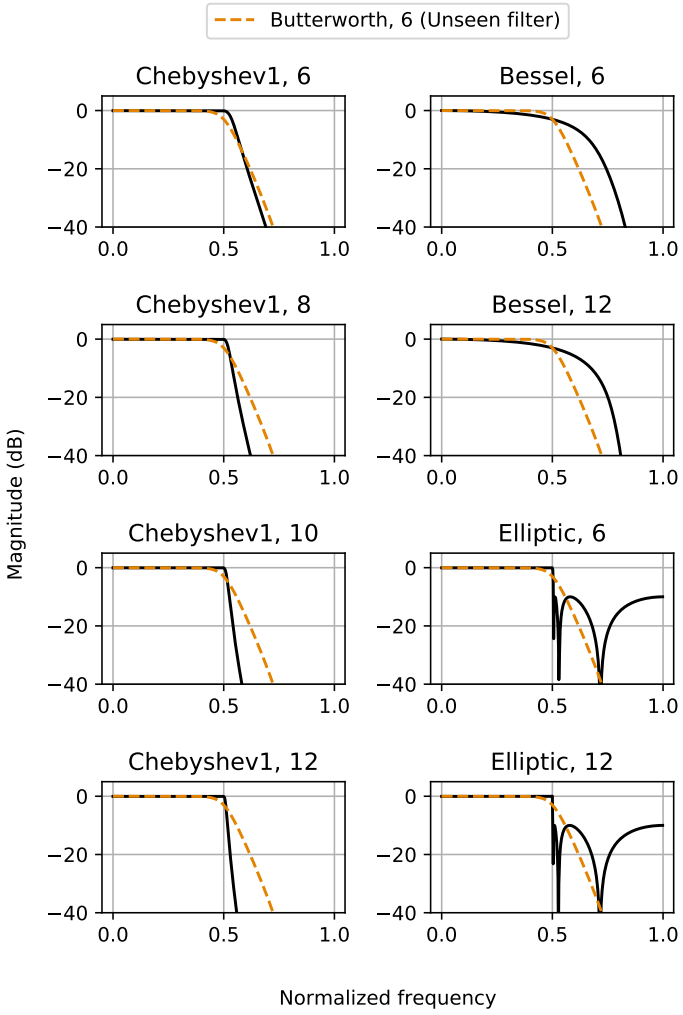


Figure 7.2: Frequency responses of the training filters. The frequency response of the unseen filter, 6th order Butterworth is superimposed on each plot.

7.4 Evaluation

We present our evaluation methods for validation, which reveal overfitting trends during training, and for testing, which indicate final performance, along with an explanation of the metric used.

7.4.1 Metric

Since our investigation focuses more on the question of generalization rather than performance, we evaluate our experiments using a single well-established metric, the signal-to-noise ratio (SNR):

$$\mathrm{SNR}(x,\hat{x}) = 10\log_{10}\frac{||x||_2^2}{||x-\hat{x}||_2^2} \tag{7.1}$$

where x is the reference signal and $\hat{x}$ is its approximation. While calculating the 2-norms, the signals are used in their stereo forms. In the specific context of our work, we consider SNR to be an appropriate choice to investigate overfitting since our models are trained with the mean-squared loss, and minimizing it corresponds to maximizing SNR.

To provide additional insight into performance, we evaluate the perceptual quality of the output audio samples, using the VGG distance, as used recently Y. Li et al. (2020) for the evaluation of music enhancement. The VGG distance between two audio samples is defined as the distance between their embeddings created by the *VGGish* network pretrained on audio classification (Hershey et al., 2017). Manocha et al. (2020) shows that the distance between deep embeddings correlates better to human evaluation, compared to hand-crafted metrics such as Perceptual Evaluation of Speech Quality (PESQ) (Rix et al., 2001) and the Virtual Speech Quality Objective Listener (ViSQOL) (Hines et al., 2012), across various audio enhancement tasks including bandwidth extension. The *VGGish* embeddings are also used in measuring the Fréchet Audio Distance (FAD), a state-of-the-art evaluation method to assess the perceptual quality of a collection of output samples (Kilgour et al., 2019). However, because FAD is used to compare two collections rather than individual audio signals, it is not applicable in our case.

To obtain the VGG embeddings, we used the *VGGish* network's open-source implementation[2]. We used the default parameters except setting the sampling frequency to 44100 Hz and the maximum frequency to 22050 Hz. In contrast to the SNR calculation, the reference implementation downmixes the stereo signals to mono before calculating the VGG embeddings. After post-processing, the embeddings take values from 0 to 255. Similar to Manocha et al. (2020), we employ the mean absolute distance to define the VGG distance as:

$$\mathrm{VGG}(x,\hat{x}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{7.2}$$

where x is the reference signal, $\hat{x}$ is its approximation; y and $\hat{y}$ are their embeddings, respectively. n is the size of the embedding tensors, which depends on the length of x.

7.4.2 Testing

To assess the overall performance of our models, we perform testing once, at the end of the training. The test split of the DSD100 dataset is reserved for our testing stage. Due to GPU memory limitations, our networks cannot process full-length songs in a single forward pass, hence they process non-overlapping chunks of audio and the outputs are later concatenated to create full-length output songs. For both VGG distance and SNR, we calculate them at the song level first, based on these full-length songs, and then take the mean over the data split to obtain the final test values.

To evaluate filter generalization, we perform two tests for each model, using seen and unseen filters. As summarized in Table 7.1, the 6th order Butterworth filter is selected as the unseen filter, as it is not used in any training setting. The seen filter only includes 6th order Chebyshev-1, as this is the only filter common to both single and multi-filter training settings.

[2]https://github.com/tensorflow/models/tree/master/research/audioset/vggish

7.4.3 Validation

To observe generalization or overfitting throughout training, we perform validation repeatedly, once every 2500 training iterations. We perform validation on 8-second audio excerpts, starting from the 8th second of each song, for only eight songs. These eight songs are the last 8 of the DSD100 training split. Since the validation is performed repeatedly throughout training, we keep the validation set sample size small. We believe that this small sample size is sufficient, because validation is only used to observe the progression of training, and the final performance evaluation is done in the testing stage. Final SNR values are calculated first by calculating the SNR over each 8 s excerpt, and then averaging over the validation songs.

Similar to testing, the validation is also performed twice, using seen and unseen filters. Validation with the unseen filter uses the 6th order Butterworth filter, as in testing. Because validation with the seen filter(s) is done to observe the training progression of each model and not to compare different models, the filters employed are the same as those in the corresponding training setting. As seen in Table 7.1, in the single-filter setting, validation with the seen filter only has the Chebyshev-1 filter, and in the multi-filter setting, it uses all eight training filters, with each assigned to processing a different song in the validation data split.

7.5 Implementation details

We build and train our models using the Pytorch framework Paszke et al., 2019 and a single Nvidia GeForce GTX 1080 Ti GPU. The model weights are initialized randomly with values drawn from the normal distribution with zero mean and unit variance. The batch size is 8. We use the Adam optimizer Kingma and Ba, 2015 with an initial learning rate of 5e-4, and with beta values 0.9 and 0.999. The learning rate is halved when the training loss reaches a plateau. We record the average training loss every 2500 iterations, and consider a plateau to correspond to no decrease in loss for 5 such consecutive measurements. Training samples are created by first randomly picking an audio file from the training dataset and then, at a random location in the audio file, extracting a chunk of stereo audio, with a length of 8192 samples, corresponding to 186 milliseconds. However, since all our models are fully-convolutional, they can process audio signals with arbitrary lengths. We train our models until convergence and for testing we use the model weights taken from the conclusion of the training.

7.6 Results

7.6.1 Validation Data

Figure 7.3 provides a high-level overview of the performance of all of the different models and training schemes, with the SNR as a function of the training iterations. While the horizontal dashed lines indicate a baseline of input SNR levels, the rest of the lines denote output SNRs for both validation settings. The SNR levels of the input validation with seen filter(s) are different

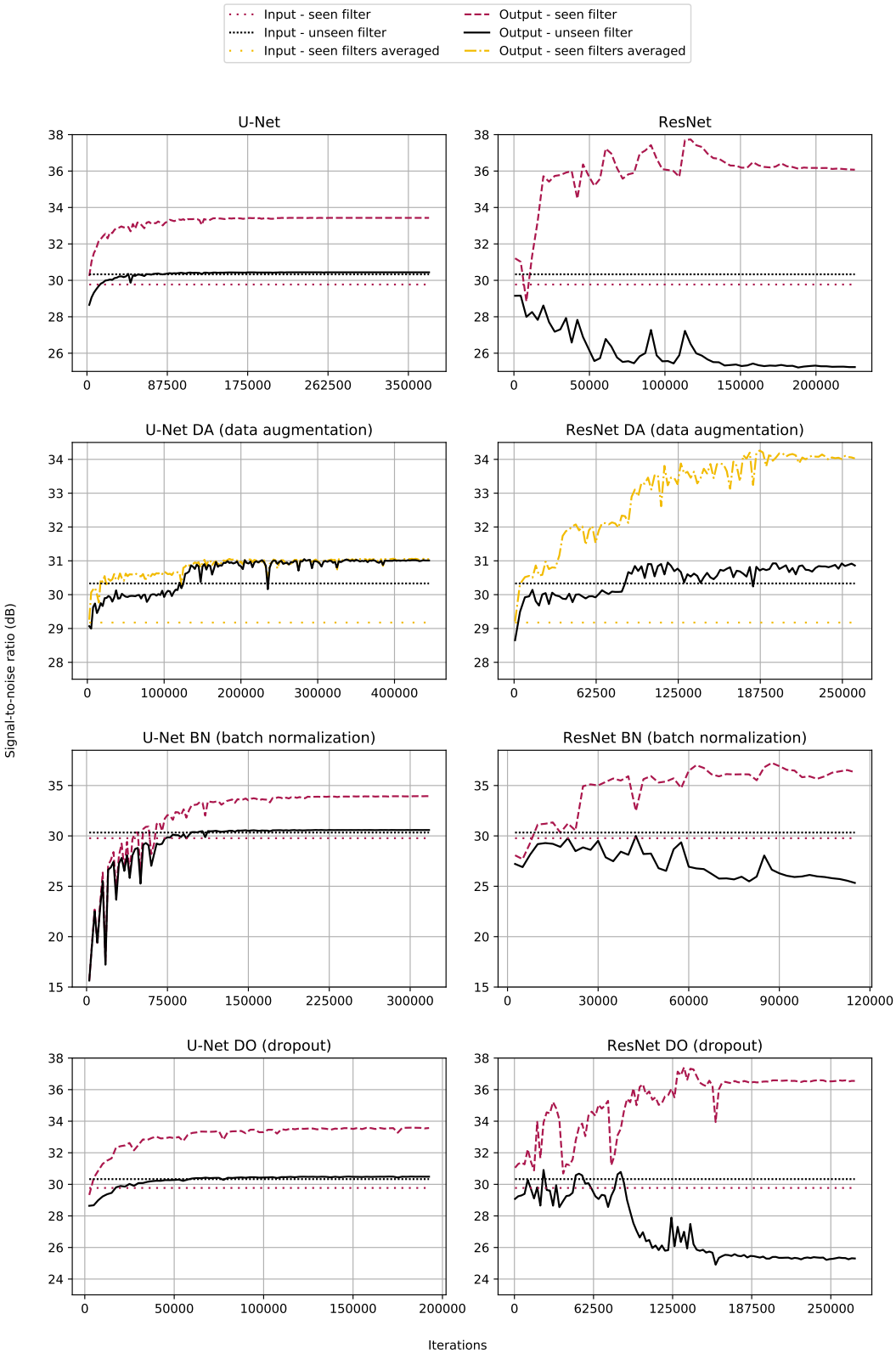


Figure 7.3: Validation performance of our models throughout training.

for the experiments with data augmentation since a different number of training filters are used as summarized in Table 7.1, and as shown in Figure 7.2 their differing frequency responses naturally lead to different baseline SNRs.

Examining the first row of Figure 7.3 we see for both networks, when the input filter is known, then large improvements in SNR over the baseline are possible. However, contrasting the U-Net with the ResNet we see performance with the unseen filter is markedly different. For the U-Net the output SNR converges to the baseline, but for the ResNet performance degrades as training continues. In this way, we see quite clear evidence of a lack of filter generalization in both models.

Moving to the second row, where training includes data augmentation, we can observe a different pattern, where both networks can improve upon the baseline SNR for the unseen filter. Contrasting the U-Net with the ResNet, we can see the ResNet offers a greater improvement upon the set of seen filters than the U-Net, albeit for approximately the same number of parameters.

Inspection of the third and fourth rows which include the two regularization techniques, we can observe a largely similar pattern to the first row, whereby the U-Net again converges to the input SNR, and the ResNet results in a lower SNR than the input. In summary, we see that for both networks, it is only training with data augmentation that we are able to find any improvement in SNR over the input.

7.6.2 Testing Data

As described in Section 7.4.3, the validation dataset is small, and the results shown in Figure 7.3 are calculated and averaged across short excerpts of 8 s in duration. In Table 7.2, we present the performance of our models on the testing data, which includes the measurement of the SNR and the VGG distance as a perceptual measure, across the entire duration of the test dataset. ΔSNR and $-\Delta$VGG represent the improvements with respect to the inputs. For SNR, ΔSNR and $-\Delta$VGG higher is better and for VGG lower is better. DA, BN, and DO correspond to data augmentation, batch normalization, and dropout, respectively. The value range of the VGG embeddings and the VGG distances is 0 to 255.

When testing with the seen filter, the ResNet models without data augmentation outperform all variants of U-Net by at least 4 dB, achieving more than a 7 dB improvement over the input SNR. The best performing model is ResNet with dropout, improving upon the input SNR by 7.4 dB. We also observe that the inclusion of data augmentation reduces performance when evaluated using the seen filter.

When testing with the unseen filter, the two best performing models use our proposed data augmentation method. Here, the ResNet variants without data augmentation produce output SNR levels which are well below those of the input. Their output SNRs are around 5.3 dB lower than the input, and around 6.6 dB lower than their U-Net based counterparts. The addition of data augmentation improves the performance of both the baseline U-Net and ResNet. Although this improvement is marginal for the U-Net, at 0.45 dB, for the ResNet, we observe a much larger improvement of 7.2 dB. In testing with the unseen filter, the best performing model is the ResNet with data augmentation, which improves upon the input SNR by 1.8 dB.

Table 7.2: Output signal-to-noise ratio (SNR) and absolute VGG distances (VGG) on the test dataset, and their improvements with respect to the inputs.

Filter	Experiment	SNR	ΔSNR	VGG	−ΔVGG
Chebyshev1−6 (seen filter)	Input	27.86		46.55	
	U-Net	30.34	+2.47	41.04	+5.51
	U-Net DA	29.78	+1.91	44.29	+2.26
	U-Net BN	30.90	+3.03	41.52	+5.03
	U-Net DO	30.49	+2.62	41.51	+5.04
	ResNet	34.94	+7.08	39.02	+7.53
	ResNet DA	30.48	+2.62	40.11	+6.43
	ResNet BN	34.37	+6.50	39.41	+7.14
	ResNet DO	**35.27**	**+7.41**	**37.23**	**+9.32**
Butterworth−6 (unseen filter)	Input	27.37		47.11	
	U-Net	28.55	+1.18	41.90	+5.21
	U-Net DA	29.00	+1.63	44.80	+2.31
	U-Net BN	28.77	+1.40	42.06	+5.06
	U-Net DO	28.62	+1.24	42.34	+4.78
	ResNet	21.96	−5.41	47.12	−0.01
	ResNet DA	**29.16**	**+1.78**	**40.52**	**+6.59**
	ResNet BN	23.23	−4.14	46.38	+0.73
	ResNet DO	22.10	−5.27	46.15	+0.96

Considering the VGG distances, the results of the U-Net variants do not change much across different filters. Compared to the seen filter setting, the ResNet variants without data augmentation exhibit worse results with the unseen filter, however, these values are very close to the input value, hence the filter overfitting in terms of the VGG distance is not as severe as the SNR. For the unseen filter setting, while the incorporation of data augmentation worsens the VGG distance by 2.8 for U-Net, it produces a much larger improvement of 6.6 for ResNet, making ResNet with data augmentation the best performing model in terms of VGG distance and SNR.

Quantitative results for each test song, along with three audio excerpts can be found online[3].

7.6.3 Sample Overfitting

In Table 7.3 we present the performance of our baseline models, without any regularization method, on the training and testing data splits separately, and evaluated on all samples in the data splits, across their full duration. Inputs are created using the low-pass filter which was also used during training (the seen filter, 6th order Chebyshev-1). To provide insight into whether any sample overfitting is occurring (i.e., that the networks are somehow memorizing the audio content of the training data) we use the seen filter, the 6th order Chebyshev-1, during testing. For both the baseline U-Net and ResNet, between training and testing data splits, the SNR improvement over the input suggests no overfitting to the audio samples themselves.

[3] https://serkansulun.com/bwe

Table 7.3: Output signal-to-noise ratio (SNR) of our baseline models, without any regularization method, on training and testing data splits separately, and their comparison to the input SNR.

Data split	Experiment	SNR (dB)	Improvement over input (dB)
Training	Input	25.99	
	U-Net	28.34	+2.35
	ResNet	33.00	+7.01
Testing	Input	27.86	
	U-Net	30.34	+2.48
	ResNet	34.94	+7.08

7.6.4 U-Net vs ResNet: Model Comparison

As stipulated in Section 7.1, we allow both the U-Net and ResNet to have a similar number of parameters. However, we informally observed a distinct difference in training time. In Table 7.4, we show several objective properties of these networks, namely the number of parameters, number of multiply-accumulate operations (MACs), and runtimes of our baseline models. Number of MACs roughly correspond to half of the number of floating point operations (FLOP). Runtime rate is the time spent in seconds, to process a signal with length of one second, during testing, i.e., a forward pass where no gradients are calculated. While both models have roughly the same number of parameters, we see that the U-Net has a much lower runtime and fewer MACs. This is due to its autoencoder-like shape, in which the convolutional layers with more channels are near the bottleneck of the network, where the spatial activation size is the smallest, effectively reducing the number of MACs and the runtime. Looking again at Figure 7.3, we can speculate that the the ResNet has greater representation power than the U-Net, as shown by its ability to better model multiple known filters than the U-Net, albeit at the cost of slower training and inference.

Table 7.4: Number of parameters, number of multiply-accumulate operations (MACs) and runtimes of our models.

Model	Number of parameters	Number of MACs	Runtime rate
U-Net	56.4M	415.3G	0.14
ResNet	55.1M	3609.4G	1.06

7.6.5 Visualization of bandwidth extension

While our proposed method operates entirely in the time-domain, we provide a graphical overview of the outputs of the two networks contrasting the baseline versions with the inclusion of data augmentation for both seen and unseen filters. To this end, we illustrate the spectrograms of one audio excerpt from the test set under each of these conditions in Figure 7.4. The inspection of the figure reveals quite different behavior of the U-Net compared to the ResNet. In general, we can observe more prominent high-frequency information in the output of the ResNet. Of particular note, is the

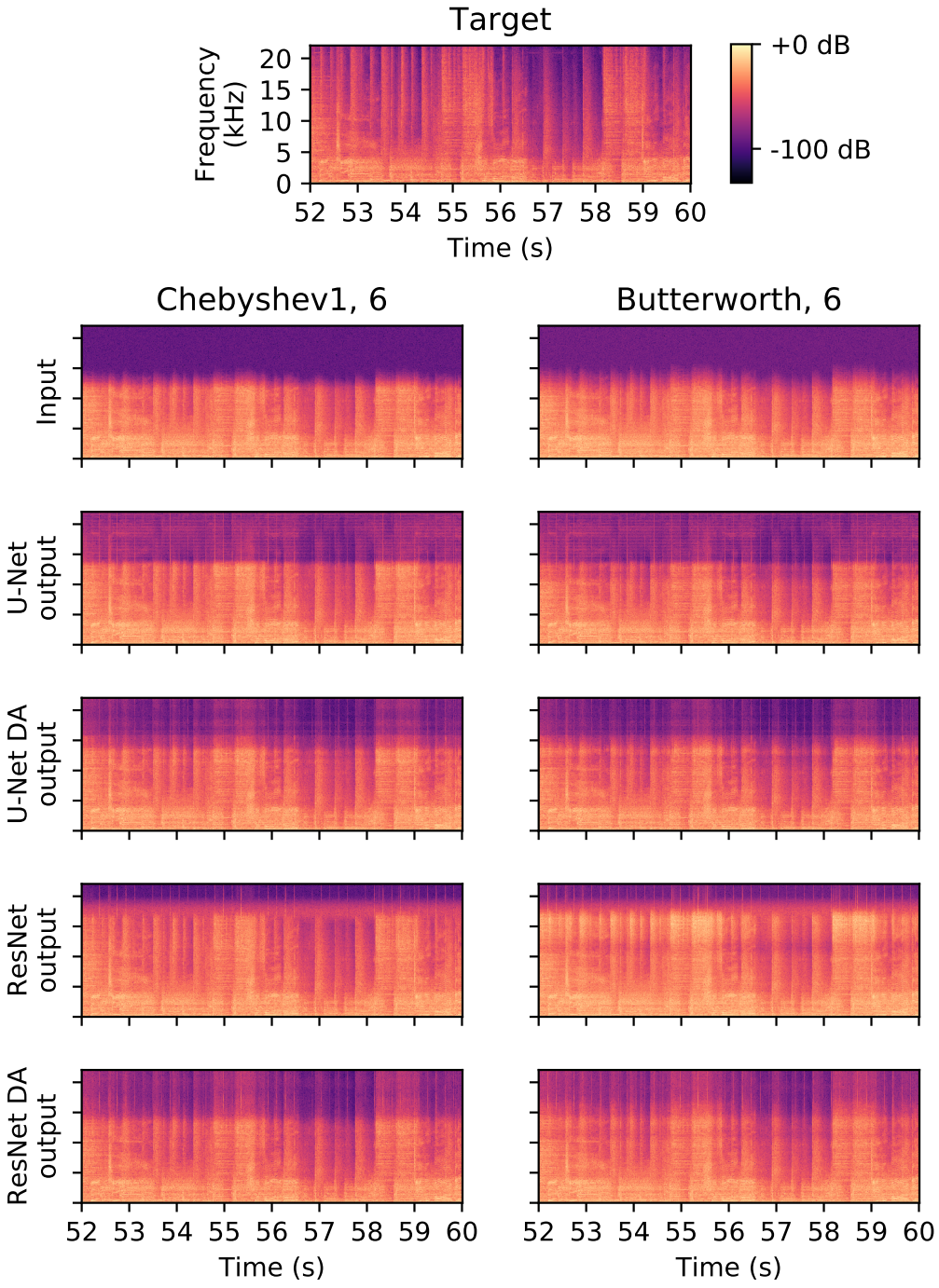


Figure 7.4: Spectograms of sample audio segments.

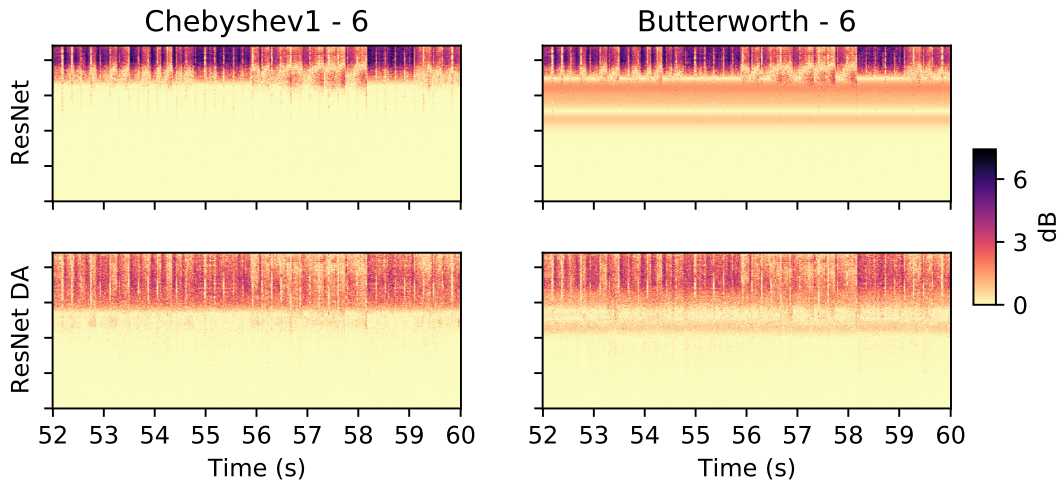


Figure 7.5: Absolute difference with respect to the target spectrogram. The colormap is inverted for better visibility.

frequency region between approximately 12-17 kHz for baseline ResNet, and the unseen Butterworth filter, which, contrasting with the target, appears to have "over-enhanced" this region. By contrast, once the data augmentation is included, this high-frequency boosting is no longer evident. To emphasize this phenomenon further, in Figure 7.5 we display the absolute difference with respect to the target spectrogram, for the baseline ResNet and ResNet with data augmentation. For the unseen Butterworth filter, in the upper half of the spectrogram, the absolute difference of the ResNet with data augmentation is much smoother compared to the baseline ResNet. In this visual representation, we can clearly observe that under all conditions the lower part of the absolute difference spectrogram is essentially unchanged, which reflects the direct additive connection of the input to the output in the network architectures.

7.7 Discussion

We raise the issue of filter generalization for deep neural networks applied to musical audio bandwidth extension. Contrary to many problems for which deep learning is used, we do not find any evidence of overfitting to audio samples themselves (i.e. the training data), but rather, we observe a clear trend for state of the art DNNs to overfit to filter shapes. When these DNNs are presented with audio excerpts that have been pre-processed with low-pass filters not included in the training, then no meaningful extension of the bandwidth can be obtained. Furthermore, the use of widely adopted regularization layers such as batch normalization and dropout fall short in alleviating this problem. Looking to the wider context and long term goal of musical audio bandwidth extension for audio enhancement, we believe that filter overfitting is a critical issue worthy of continued focus.

To address the filter overfitting issue we propose a novel data augmentation approach, which uses multiple filters at the time of training. Our results demonstrate that without data augmentation, filter overfitting increases as training progresses, whereas including data augmentation is a promising step towards achieving filter generalization. While the improvement in generalization for the U-Net is quite small, a more pronounced effect can be observed for the ResNet, which retains high performance across multiple seen training filters. It is particularly noteworthy that the ResNet variants without data augmentation produce very poor results when tested with an unseen filter, with output quality well below that of the input. In this way, the incorporation of data augmentation was the only means to achieve SNR levels that are above the input.

In addition to the primary findings concerning filter generalization, this is, to the best of our knowledge, the first comparison between U-Nets and ResNets in the field of audio processing, and perhaps the first ever comparison of these approaches given a similar number of parameters. Examining the results of testing with the seen filter, we observe that baseline ResNet outperforms baseline U-Net by a large margin. However, when testing with the unseen filter, baseline ResNet performs worst.

We argue that the ResNet has more representation power than the U-Net because while the U-Net reduces the spatial activation sizes in its downsampling blocks, the ResNet keeps the spatial activation sizes constant, starting from its input until its output, thus minimizing the loss of information. Even though the networks have the same number of parameters, we can quantify this higher representation power by comparing the number of multiply-accumulate operations. This higher representation power results in the ResNet performing much better in tests with the seen filter while demonstrating much higher levels of filter overfitting when there is no data augmentation. We show that using the proposed data augmentation method, this powerful network can be successfully regularized, and achieves the best SNR when testing with the unseen filter. While we chose to keep the number of parameters within the two models roughly equal, we note that compared to the ResNet, the U-Net is 7.5 times faster and does nearly 9 times fewer multiply-accumulate operations (MACs). In this way, the U-Net may be a preferred architecture for real-time streaming applications.

We now move on to the conclusion, where we summarize the key findings and implications of our work and highlight potential future directions.

Chapter 8

Conclusion

In this thesis, we have presented a novel approach for video-based music generation, creating EMSYNC—an innovative framework that synchronizes music with video by aligning both emotional content and temporal boundaries. By integrating video analysis, emotion-based conditioning, and temporal boundary conditioning into a single music generation pipeline, EMSYNC provides a solution to key challenges in generating emotionally and rhythmically aligned music for video.

8.1 Contributions

One of the major contributions of this work is the creation of models for emotion classification of arbitrary videos and genre classification of cinematic trailers. These models are key to the success of our video-based music generation system, as they provide the emotional context and genre-specific features necessary for music generation. Our emotion classifier, VEMOCLAP, is capable of identifying the emotional tone of arbitrary videos, making it a versatile tool for various applications beyond music generation. Similarly, our genre classifier for cinematic trailers leverages pretrained models for feature extraction and allows for efficient, high-quality genre classification. These pretrained models are valuable resources for the community and have wide-ranging potential in multimedia content analysis, including automatic content categorization, sentiment analysis, and recommendation systems.

Another significant contribution is our music generator, which is the first to be conditioned on continuous-valued conditions—specifically, valence and arousal values—rather than discrete emotion categories. This breakthrough enables much finer control over the generated music, as it allows for the precise adjustment of emotional expression and musical dynamics. By avoiding the discretization of emotion features, we preserve the continuous nature of emotional expression, resulting in music that is more nuanced and reflective of the emotional complexity in the video content. This continuous-conditioning approach offers broader flexibility for fine-tuning music generation, allowing for more expressive and adaptive outputs that better align with varying emotional tones across different videos. Furthermore, this approach has implications for other

sequence generation models, particularly in areas like natural language processing, where fine-grained control over generated content could enhance coherence and emotional depth in generated sequences.

We also created the first large-scale, emotion-labeled MIDI dataset specifically designed for music generation. By combining emotional annotations derived from the Spotify Developers API and song lyrics, this dataset provides a powerful resource for training emotion-based music generators. The impact of this emotion-labeled dataset is profound, as it bridges a gap in the field of symbolic music generation by providing emotion-rich training data that allows the generation of music explicitly aligned with emotional states. The dataset opens up new opportunities for research in affective music generation, enabling the creation of systems capable of producing music that responds dynamically to emotional inputs in various contexts such as films, games, and interactive media.

Furthermore, the emotion-labeled MIDI datasets created as part of this work have broader implications for the field of artificial intelligence and music. The dataset is a unique resource for training systems that not only generate music but also understand and adapt to the emotional qualities of a piece. By providing both continuous emotion labels and symbolic music data, we have set the stage for the investigation of more advanced music generation systems that can respond to emotional cues in a natural and seamless manner.

Our work also makes an important contribution by introducing a system for synchronizing video content with music through temporal boundary conditioning. By aligning musical events with video scene cuts, our approach ensures that the music is not only emotionally aligned but also rhythmically synchronized with the video's key moments. This synchronization enhances the overall multimedia experience, particularly in applications such as film scoring, advertisements, and video games, where precise timing is crucial.

While MIDI events are not inherently aligned to a fixed time grid, our approach nonetheless enables effective temporal conditioning of the generator. This has broader implications for sequence processing tasks that involve transformers, as it demonstrates a new way of integrating temporal boundaries into generative models. This methodology can extend to various applications, such as text generation, speech synthesis, and other sequence-based tasks, where precise temporal alignment is crucial for maintaining coherence and enhancing the overall output.

In summary, EMSYNC represents a significant step forward in the field of video-based music generation. It is the first system to integrate emotion-based and boundary-based conditioning for music generation, enabling the creation of music that is both emotionally rich and rhythmically synchronized with video. The pretrained emotion and genre classifiers, the continuous-conditioning approach to music generation, and the emotion-labeled MIDI dataset are all major innovations that pave the way for more advanced and versatile music generation systems.

Moreover, EMSYNC outperforms the two existing methods in user studies across all subjective metrics, for both participant groups—with and without self-reported knowledge of music theory. Our method consistently ranks best in music richness, quality, emotional alignment, and timing synchronization, as confirmed by participants with varying levels of familiarity with music

theory. This demonstrates not only the technical efficiency of EMSYNC but also its potential to reshape video-based music generation, offering a more expressive and adaptive solution to synchronizing music with video content. The results of these studies further validate the practical effectiveness of EMSYNC, establishing it as a notable advancement in this field.

Finally, we provided a preliminary analysis of audio generation through the audio bandwidth extension task. We highlighted a significant issue with the use of synthetic data, which is common in many works in the field of audio generation. Specifically, we demonstrated that models trained with one data synthesis method do not generalize well to data created by other synthesis methods. To address this, we took the first step towards mitigating this problem by diversifying the data synthesis methods to better approximate real-world scenarios. We believe that the insights from this work are valuable for other audio generation tasks, including MIDI-to-audio.

We make every stage of our research accessible to the community through open-source code, datasets, pretrained models, and interactive demos. By releasing these resources, we aim to foster collaboration and innovation not only within the research community but also for the general public. This openness allows other researchers to build upon our work, test new ideas, and advance the field of video-based music generation. Additionally, providing access to these tools empowers practitioners, developers, and enthusiasts to explore and apply our methods in diverse real-world contexts, from film production to game development, and beyond. Through this contribution, we hope to inspire further exploration and progress in both the academic and creative sectors.

Our major contributions in this thesis are summarized below.

- We provide emotion labels in the form of valence-arousal values for 34,791 songs in the Lakh MIDI dataset (Raffel, 2016), creating the largest existing MIDI dataset with emotion labels.
- We present the first MIDI generator that can be conditioned on continuous valence-arousal values.
- We create the first MIDI generator that can be conditioned on temporal boundaries in a direct manner.
- Our cinematic trailer genre classifier outperforms all existing methods on MovieNet, the largest cinematic trailer dataset (Q. Huang et al., 2020).
- Our video emotion classifier outperforms all existing methods on Ekman-6, the largest user-generated (arbitrary) video dataset with emotion labels (B. Xu et al., 2018).
- Our video-based music generator outperforms all existing methods in user studies, across all metrics and demographics.
- We demonstrate that training audio generation models with synthetic data leads to generalization issues, and we take initial steps to address this challenge.

We list the papers we published throughout our research below.

- **S. Sulun**, M. E. P. Davies, and P. Viana, "Symbolic Music Generation Conditioned on Continuous-Valued Emotions," IEEE Access, vol. 10, pp. 44617–44626, 2022.
- **S. Sulun**, P. Oliveira, and P. Viana, "Emotion4MIDI: A Lyrics-Based Emotion-Labeled Symbolic Music Dataset," in Progress in Artificial Intelligence, Cham: Springer Nature Switzerland, 2023, pp. 77–89.
- **S. Sulun**, P. Viana, and M. E. P. Davies, "Video Soundtrack Generation by Aligning Emotions and Temporal Boundaries," arXiv: arXiv:2502.10154.
- **S. Sulun**, P. Viana, and M. E. P. Davies, "VEMOCLAP - A video emotion classification web application," IEEE International Symposium on Multimedia (ISM), 2024.
- **S. Sulun**, P. Viana, and M. E. P. Davies, "Movie trailer genre classification using multimodal pretrained features," Expert Systems with Applications, vol. 258, p. 125209, 2024.
- **S. Sulun** and M. E. P. Davies, "On filter generalization for music bandwidth extension using deep neural networks," IEEE Journal of Selected Topics in Signal Processing, vol. 15, no. 1, pp. 132–142, 2020.

8.2 Future directions

Looking ahead, we aim to offer insight and inspiration to future researchers in the fields of machine learning and multimedia. Throughout our research, we favored using large and relatively simple models trained on large datasets, rather than complex models trained on small datasets. The qualitative and quantitative superiority of our approach aligns with the current trend that training larger models on larger datasets yields better results (Bernstein et al., 2021; Sevilla et al., 2022; M. Tan & Le, 2019). This approach has long been adopted by the industry, with text generation models like ChatGPT[1], LLaMA[2], and Gemini[3], as well as image generation models like DALL-E[4] and Imagen[5], and audio generation models like Suno[6].

While the hardware capabilities of major tech companies give them a clear advantage, another overlooked factor is their access to large-scale proprietary datasets. Therefore, we emphasize the importance of creating newer, larger open-source, and peer-reviewed datasets, rather than focusing solely on building complex models on small datasets for incremental improvements over the existing state-of-the-art.

One effective approach to creating better datasets is by labeling existing ones. The success of image processing models like CLIP stems from the contrastive pretraining method, which pairs images with tags and other textual descriptions scraped from the internet (Radford et al., 2021).

[1] https://www.chatgpt.com
[2] https://www.llama.com
[3] https://gemini.google.com/app
[4] https://openai.com/index/dall-e-3
[5] https://deepmind.google/technologies/imagen-3
[6] https://suno.com

Similar methods could be applied to other types of multimedia. However, a key challenge with this approach is the intentional or accidental use of copyrighted material. To keep pace with industry research advances, it seems likely that academic labs will need to collaborate with legal teams or allocate research funds for legal advice, just as the industry does.

Many datasets rely on content from video-sharing platforms (Q. Huang et al., 2020; B. Xu et al., 2018). The rise of large language models has made text data particularly valuable. Newer datasets should leverage the tags and descriptions of user-uploaded videos. Contrastive learning strategies can be especially useful for extracting meaningful features from pairs of video and text (Alayrac et al., 2020).

Video emotion classification becomes more challenging when the analyzed video has a complex plot, with semantic variations and subtleties. Extracting this complex semantic information directly from raw pixels and audio would likely be inefficient, if not impossible. Our approach of using pretrained features is a step toward efficiently understanding this semantic complexity. As a next step, videos could be transcribed in detail into text using similar pretrained features, and large language models (LLMs) with reasoning capabilities (Wei et al., 2022) could process the resulting text to detect potential plot twists and nuances.

When data is scarce, using high-dimensional representations can lead to overfitting, often referred to as the *curse of dimensionality*. Our method of using pretrained features effectively serves as a feature reduction technique, condensing large raw frames into smaller, more meaningful features. Other features can certainly be added. For instance, emotion classification of speech was one aspect we planned to include but struggled to find a reasonably performing open-source model. However, even though pretrained feature extractors work in inference mode, using many of them would significantly increase runtime. To mitigate this, employing knowledge distillation to reduce the sizes of these pretrained networks would be advantageous (Hinton et al., 2015). Furthermore, combining multiple architectures into a single unified model would be a challenging yet rewarding research task. Finally, utilizing multiple datasets from different classifications through continual learning could help address data scarcity (L. Wang et al., 2024).

The data scarcity problem is also evident with labeled MIDI datasets. Our work utilizes features from the Spotify Developer API to label MIDI files, which we refer to as "weak labels" since these features were originally created for audio, not MIDI. However, our work demonstrates that even when the labels are weak, having a large dataset with these labels is still beneficial. Future researchers can gather textual features, such as descriptions or tags associated with musical audio from the internet, assign them to their respective MIDI files, and apply LLMs and/or contrastive learning methods in a similar manner.

Generating multi-instrument audio from MIDI, or vice versa, using deep neural networks (DNNs) is a relatively underexplored area. Effective audio-to-MIDI models could facilitate the synthetic creation of large MIDI datasets. On the other hand, MIDI-to-audio models with realistic outputs would enable the production of high-quality sound from symbolic music compositions. As diffusion models have already demonstrated their ability to generate images of unprecedented

quality, they could also be a promising candidate for high-quality, realistic audio generation (Sohl-Dickstein et al., 2015).

Finally, the advent of Suno certainly represents a disruptive breakthrough in the field of music generation. While their methods are proprietary, it is plausible to assume that large private datasets of music in audio format contribute significantly to their success. The presentation of an open-source, peer-reviewed method that achieves similar levels of success would mark an incredible advancement in music generation research.

References

Adlam, B., Weill, C., & Kapoor, A. (2019). Investigating under and overfitting in wasserstein generative adversarial networks. *CoRR*, *abs/1910.14137*arXiv 1910.14137. http://arxiv.org/abs/1910.14137

Adler, A., Emiya, V., Jafari, M. G., Elad, M., Gribonval, R., & Plumbley, M. D. (2012). Audio inpainting. *IEEE Transactions on Speech Audio Processing*, *20*(3), 922–932. https://doi.org/10.1109/TASL.2011.2168211

Ak, K. E., Lee, G.-G., Xu, Y., & Shen, M. (2023). Leveraging efficient training and feature fusion in transformers for multimodal classification, In *Ieee international conference on image processing, icip 2023, kuala lumpur, malaysia, october 8-11, 2023*, IEEE. https://doi.org/10.1109/ICIP49359.2023.10223098

Alayrac, J.-B., Recasens, A., Schneider, R., Arandjelovic, R., Ramapuram, J., Fauw, J. D., Smaira, L., Dieleman, S., & Zisserman, A. (2020). Self-supervised multimodal versatile networks, In *Advances in neural information processing systems*. https://proceedings.neurips.cc/paper/2020/hash/0060ef47b12160b9198302ebdb144dcf-Abstract.html

Almahairi, A., Ballas, N., Cooijmans, T., Zheng, Y., Larochelle, H., & Courville, A. C. (2016). Dynamic capacity networks, In *Proceedings of the 33nd international conference on machine learning, ICML 2016, new york city, ny, usa, june 19-24, 2016*, JMLR.org. http://proceedings.mlr.press/v48/almahairi16.html

Almeida, A., de Villiers, J. P., De Freitas, A., & Velayudan, M. (2022). The complementarity of a diverse range of deep learning features extracted from video content for video recommendation. *Expert Systems with Applications*, *192*, 116335. https://doi.org/10.1016/j.eswa.2021.116335

Almeida, J., Vilaça, L., Teixeira, I. N., & Viana, P. (2021). Emotion identification in movies through facial expression recognition. *Applied Sciences*, *11*(15). https://doi.org/10.3390/app11156827

Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lucic, M., & Schmid, C. (2021). Vivit: A video vision transformer, In *2021 IEEE/CVF international conference on computer vision, ICCV 2021, montreal, qc, canada, october 10-17, 2021*, IEEE. https://doi.org/10.1109/ICCV48922.2021.00676

Ba, L. J., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *CoRR*, *abs/1607.06450*arXiv 1607.06450. http://arxiv.org/abs/1607.06450

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate, In *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings*. http://arxiv.org/abs/1409.0473

Bain, M., Nagrani, A., Brown, A., & Zisserman, A. (2020). Condensed movies: Story based retrieval with contextual embeddings, In *Computer vision - ACCV 2020 - 15th asian conference on computer vision, kyoto, japan, november 30 - december 4, 2020, revised selected papers, part V*, Springer. https://doi.org/10.1007/978-3-030-69541-5_28

Bansal, D., Raj, B., & Smaragdis, P. (2005). Bandwidth expansion of narrowband speech using non-negative matrix factorization, In *9th european conference on speech communication and technology, interspeech-eurospeech 2005, lisbon, portugal, september 4-8, 2005*, ISCA. https://doi.org/10.21437/INTERSPEECH.2005-528

Barsoum, E., Zhang, C., Ferrer, C. C., & Zhang, Z. (2016). Training deep networks for facial expression recognition with crowd-sourced label distribution, In *Proceedings of the 18th acm international conference on multimodal interaction*, Tokyo Japan, ACM. https://doi.org/10.1145/2993148.2993165

Bauer, P., & Fingscheidt, T. (2008). An hmm-based artificial bandwidth extension evaluated by cross-language training and test, In *Proceedings of the IEEE international conference on acoustics, speech, and signal processing, ICASSP 2008, march 30 - april 4, 2008, caesars palace, las vegas, nevada, USA*, IEEE. https://doi.org/10.1109/ICASSP.2008.4518678

Berg, J., & Wingstedt, J. (2005). Relations between selected musical parameters and expressed emotions: Extending the potential of computer entertainment, In *Proceedings of the international conference on advances in computer entertainment technology, ace 2005, valencia, spain, june 15-15, 2005*, ACM. https://doi.org/10.1145/1178477.1178499

Bernstein, L., Sludds, A., Hamerly, R., Sze, V., Emer, J., & Englund, D. (2021). Freely scalable and reconfigurable optical hardware for deep learning. *Scientific reports*, *11*(1), 3144. https://doi.org/10.1038/s41598-021-82543-3

Bertin-Mahieux, T., Ellis, D. P. W., Whitman, B., & Lamere, P. (2011). The million song dataset, In *Proceedings of the 12th international society for music information retrieval conference, ISMIR 2011, miami, florida, usa, october 24-28, 2011*, University of Miami. http://ismir2011.ismir.net/papers/OS6-1.pdf

Bharucha, J., & Krumhansl, C. L. (1983). The representation of harmonic structure in music: Hierarchies of stability as a function of context. *Cognition*, *13*(1), 63–102. https://doi.org/10.1016/0010-0277(83)90003-3

Bieda, I., Kisil, A., & Panchenko, T. (2021). An approach to scene change detection, In *2021 11th IEEE international conference on intelligent data acquisition and advanced computing systems: Technology and applications (idaacs), cracow, poland, september 22-25, 2021*, IEEE. https://doi.org/10.1109/IDAACS53288.2021.9660887

Bittner, R. M., Wilkins, J., Yip, H., & Bello, J. P. (2016). Medleydb 2.0: New data and a system for sustainable data collection. *ISMIR Late Breaking and Demo Papers*. https://bpb-us-e1.wpmucdn.com/wp.nyu.edu/dist/2/2294/files/2016/08/bittner-medleydb.pdf?bid=2294

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. arXiv. https://doi.org/10.48550/arXiv.2004.10934

Bose, D., Hebbar, R., Somandepalli, K., Zhang, H., Cui, Y., Cole-McLaughlin, K., Wang, H., & Narayanan, S. (2023). MovieCLIP: Visual scene recognition in movies, In *IEEE/CVF winter conference on applications of computer vision, WACV 2023, waikoloa, hi, usa, january 2-7, 2023*, IEEE. https://doi.org/10.1109/WACV56688.2023.00212

Briot, J.-P., Hadjeres, G., & Pachet, F.-D. (2020). *Deep learning techniques for music generation*. Springer. https://doi.org/10.1007/978-3-319-70163-9

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language

models are few-shot learners, In *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, neurips 2020, december 6-12, 2020, virtual*. https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

Bucila, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression, In *Proceedings of the twelfth ACM SIGKDD international conference on knowledge discovery and data mining, philadelphia, pa, usa, august 20-23, 2006*, ACM. https://doi.org/10.1145/1150402.1150464

Buechel, S., & Hahn, U. (2017). Emobank: Studying the impact of annotation perspective and representation format on dimensional emotion analysis, In *Proceedings of the 15th conference of the european chapter of the association for computational linguistics, EACL 2017, valencia, spain, april 3-7, 2017, volume 2: Short papers*, Association for Computational Linguistics. https://doi.org/10.18653/V1/E17-2092

Buhler, J. (2018). *Theories of the soundtrack*. Oxford University Press. https://books.google.com/books?id=n_NwDwAAQBAJ&lpg=PP1&ots=RKwzh_dMet&lr=&pg=PP1&redir_esc=y#v=onepage&q&f=false

Calvo, R. A., & Mac Kim, S. (2013). Emotions in text: Dimensional and categorical models. *Computational Intelligence*, *29*(3), 527–543. https://doi.org/10.1111/j.1467-8640.2012.00456.x

Camacho-Collados, J., Rezaee, K., Riahi, T., Ushio, A., Loureiro, D., Antypas, D., Boisson, J., Anke, L. E., Liu, F., & Cámara, E. M. (2022). Tweetnlp: Cutting-edge natural language processing for social media, In *Proceedings of the the 2022 conference on empirical methods in natural language processing, EMNLP 2022 - system demonstrations, abu dhabi, uae, december 7-11, 2022*, Association for Computational Linguistics. https://doi.org/10.18653/V1/2022.EMNLP-DEMOS.5

Canny, J. F. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis Machine Intelligence*, *8*(6), 679–698. https://doi.org/10.1109/TPAMI.1986.4767851

Carreira, J., & Zisserman, A. (2017). Quo vadis, action recognition? a new model and the kinetics dataset, In *2017 ieee conference on computer vision and pattern recognition, cvpr 2017, honolulu, hi, usa, july 21-26, 2017*, IEEE Computer Society. https://doi.org/10.1109/CVPR.2017.502

Cascante-Bonilla, P., Sitaraman, K., Luo, M., & Ordonez, V. (2019). Moviescope: Large-scale analysis of movies using multiple modalities. *arXiv preprint arXiv:1908.03180*arXiv 1908.03180. https://doi.org/10.48550/arXiv.1908.03180

Chatterjee, A., Narahari, K. N., Joshi, M., & Agrawal, P. (2019). Semeval-2019 task 3: Emocontext contextual emotion detection in text, In *Proceedings of the 13th international workshop on semantic evaluation, semeval@naacl-hlt 2019, minneapolis, mn, usa, june 6-7, 2019*, Association for Computational Linguistics. https://doi.org/10.18653/V1/S19-2005

Chen, C., Wu, Z., & Jiang, Y.-G. (2016). Emotion in context: Deep semantic feature fusion for video emotion recognition, In *Acm-mm 2016*, ACM. https://doi.org/10.1145/2964284.2967196

Chen, K., Wang, C.-i., Berg-Kirkpatrick, T., & Dubnov, S. (2020). Music sketchnet: Controllable music generation via factorized representations of pitch and rhythm, In *Proceedings of the 21th international society for music information retrieval conference, ISMIR 2020, montreal, canada, october 11-16, 2020*. http://archives.ismir.net/ismir2020/paper/000146.pdf

Chen, S., Wu, Y., Wang, C., Liu, S., Tompkins, D., Chen, Z., Che, W., Yu, X., & Wei, F. (2023). Beats: Audio pre-training with acoustic tokenizers, In *International conference on machine learning, ICML 2023, 23-29 july 2023, honolulu, hawaii, USA*, PMLR. https://proceedings.mlr.press/v202/chen23ag.html

Cheng, Y. M., O'Shaughnessy, D. D., & Mermelstein, P. (1994). Statistical recovery of wideband speech from narrowband speech. *IEEE Transactions on Speech Audio Processing*, *2*(4), 544–548. https://doi.org/10.1109/89.326637

Chennoukh, S., Gerrits, A. J., Miet, G., & Sluijter, R. J. (2001). Speech enhancement via frequency bandwidth extension using line spectral frequencies, In *IEEE international conference on acoustics, speech, and signal processing, ICASSP 2001, 7-11 may, 2001, salt palace convention center, salt lake city, utah, usa, proceedings*, IEEE. https://doi.org/10.1109/ICASSP.2001.940919

Chiou, E., Giganti, F., Bonet-Carne, E., Punwani, S., Kokkinos, I., & Panagiotaki, E. (2018). Prostate cancer classification on VERDICT DW-MRI using convolutional neural networks, In *Machine learning in medical imaging - 9th international workshop, MLMI 2018, held in conjunction with MICCAI 2018, granada, spain, september 16, 2018, proceedings*, Springer. https://doi.org/10.1007/978-3-030-00919-9_37

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014a). Learning phrase representations using rnn encoder-decoder for statistical machine translation, In *Proceedings of the 2014 conference on empirical methods in natural language processing, emnlp 2014, october 25-29, 2014, doha, qatar, a meeting of sigdat, a special interest group of the acl*. https://doi.org/10.3115/v1/d14-1179

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014b). On the properties of neural machine translation: Encoder-decoder approaches, In *Proceedings of ssst@emnlp 2014, eighth workshop on syntax, semantics and structure in statistical translation, doha, qatar, 25 october 2014*, Association for Computational Linguistics. https://doi.org/10.3115/V1/W14-4012

Choi, K., Fazekas, G., & Sandler, M. B. (2016). Automatic tagging using deep convolutional neural networks, In *Proceedings of the 17th international society for music information retrieval conference, ISMIR 2016, new york city, united states, august 7-11, 2016*. https://wp.nyu.edu/ismir2016/wp-content/uploads/sites/2294/2016/07/009%5C_Paper.pdf

Choi, K., Hawthorne, C., Simon, I., Dinculescu, M., & Engel, J. H. (2020). Encoding musical style with transformer autoencoders, In *Proceedings of the 37th international conference on machine learning, ICML 2020, 13-18 july 2020, virtual event*, PMLR. http://proceedings.mlr.press/v119/choi20b.html

Church, K. W., Chen, Z., & Ma, Y. (2021). Emerging trends: A gentle introduction to fine-tuning. *Nat. Lang. Eng.*, *27*(6), 763–778. https://doi.org/10.1017/S1351324921000322

Cohen, A. J. (1990). Understanding musical soundtracks. *Empirical Studies of the Arts*, *8*(2), 111–124. https://doi.org/10.2190/8Y6G-KTM8-VDX4-UHRW

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale, In *Proceedings of the 58th annual meeting of the association for computational linguistics, ACL 2020, online, july 5-10, 2020*, Association for Computational Linguistics. https://doi.org/10.18653/V1/2020.ACL-MAIN.747

Conneau, A., Rinott, R., Lample, G., Williams, A., Bowman, S. R., Schwenk, H., & Stoyanov, V. (2018). XNLI: evaluating cross-lingual sentence representations, In *Proceedings of the 2018 conference on empirical methods in natural language processing, brussels, belgium,*

october 31 - november 4, 2018, Association for Computational Linguistics. https://doi.org/10.18653/V1/D18-1269

Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., & Défossez, A. (2023). Simple and controllable music generation, In *Advances in neural information processing systems 36: Annual conference on neural information processing systems 2023, neurips 2023, new orleans, la, usa, december 10 - 16, 2023*. http://papers.nips.cc/paper%5C_files/paper/2023/hash/94b472a1842cd7c56dcb125fb2765fbd-Abstract-Conference.html

Costa-jussà, M. R., Cross, J., Çelebi, O., Elbayad, M., Heafield, K., Heffernan, K., Kalbassi, E., Lam, J., Licht, D., Maillard, J., Sun, A. Y., Wang, S., Wenzek, G., Youngblood, A., Akula, B., Barrault, L., Gonzalez, G. M., Hansanti, P., Hoffman, J., … Wang, J. (2022). No language left behind: Scaling human-centered machine translation. https://doi.org/10.48550/ARXIV.2207.04672

Cutler, R., Saabas, A., Pärnamaa, T., Purin, M., Indenbom, E., Ristea, N.-C., Guzvin, J., Gamper, H., Braun, S., & Aichner, R. (2023). ICASSP 2023 acoustic echo cancellation challenge. *CoRR*, *abs/2309.12553*arXiv 2309.12553. https://doi.org/10.48550/ARXIV.2309.12553

Cutting, J. E. (2005). Perceiving scenes in film and in the world. *Moving image theory: Ecological considerations*, 9–27. https://www.researchgate.net/profile/James-Cutting/publication/237009484_Perceiving_scenes_in_film_and_in_the_world/links/0046351ae10f2baa8c000000/Perceiving-scenes-in-film-and-in-the-world.pdf

Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., & Liu, R. (2020). Plug and play language models: A simple approach to controlled text generation, In *8th international conference on learning representations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*, OpenReview.net. https://openreview.net/forum?id=H1edEyBKDS

Defernez, M., & Kemsley, E. K. (1999). Avoiding overfitting in the analysis of high-dimensional data with artificial neural networks (ANNs). *Analyst*, *124*(11), 1675–1681. https://doi.org/10.1039/A905556H

Deldjoo, Y., Constantin, M. G., Ionescu, B., Schedl, M., & Cremonesi, P. (2018). MMTF-14K: a multifaceted movie trailer feature dataset for recommendation and retrieval, In *Proceedings of the 9th ACM multimedia systems conference, mmsys 2018, amsterdam, the netherlands, june 12-15, 2018*, ACM. https://doi.org/10.1145/3204949.3208141

Demszky, D., Movshovitz-Attias, D., Ko, J., Cowen, A. S., Nemade, G., & Ravi, S. (2020). Goemotions: A dataset of fine-grained emotions, In *Proceedings of the 58th annual meeting of the association for computational linguistics, ACL 2020, online, july 5-10, 2020*, Association for Computational Linguistics. https://doi.org/10.18653/V1/2020.ACL-MAIN.372

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database, In *2009 IEEE computer society conference on computer vision and pattern recognition (CVPR 2009), 20-25 june 2009, miami, florida, USA*, IEEE Computer Society. https://doi.org/10.1109/CVPR.2009.5206848

Deutsch, S. (2007). The soundtrack (putting music in its place). *The Soundtrack*, *1*(1), 3–13. http://eprints.bournemouth.ac.uk/1307/

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding, In *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies, NAACL-HLT 2019, minneapolis, mn, usa, june 2-7, 2019, volume 1 (long and short papers)*, Association for Computational Linguistics. https://doi.org/10.18653/v1/n19-1423

Di, S., Jiang, Z., Liu, S., Wang, Z., Zhu, L., He, Z., Liu, H., & Yan, S. (2021). Video background music generation with controllable music transformer, In *MM '21: ACM multimedia conference, virtual event, china, october 20 - 24, 2021*, ACM. https://doi.org/10.1145/3474085.3475195

Donahue, C., Mao, H. H., Li, Y. E., Cottrell, G. W., & McAuley, J. J. (2019). Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training, In *Proceedings of the 20th international society for music information retrieval conference, ISMIR 2019, delft, the netherlands, november 4-8, 2019*. http://archives.ismir.net/ismir2019/paper/000083.pdf

Dong, H.-W., Hsiao, W.-Y., Yang, L.-C., & Yang, Y.-H. (2018). Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. https://doi.org/10.1609/aaai.v32i1.11312

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale, In *9th international conference on learning representations, iclr 2021, virtual event, austria, may 3-7, 2021*, OpenReview.net. https://openreview.net/forum?id=YicbFdNTTy

Eck, D., & Schmidhuber, J. (2002). A first look at music composition using LSTM recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, *103*, 48. https://people.idsia.ch/~juergen/blues/IDSIA-07-02.pdf

Eerola, T., & Vuoskoski, J. K. (2011). A comparison of the discrete and dimensional models of emotion in music. *Psychology of Music*, *39*(1), https://doi.org/10.1177/0305735610362821, 18–49. https://doi.org/10.1177/0305735610362821

Ekman, P. (1971). Universals and cultural differences in facial expressions of emotion. *Nebraska Symposium on Motivation*, *19*, 207–283. https://psycnet.apa.org/record/1973-11154-001

Ekman, P. (1975). *Unmasking the face; a guide to recognizing emotions from facial clues*. Englewood Cliffs, N.J, Prentice-Hall. https://psycnet.apa.org/record/1975-31746-000

Engel, J. H., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., & Roberts, A. (2019). Gansynth: Adversarial neural audio synthesis, In *7th international conference on learning representations, ICLR 2019, new orleans, la, usa, may 6-9, 2019*, OpenReview.net. https://openreview.net/forum?id=H1xQVn09FX

Ens, J., & Pasquier, P. (2020). Flexible generation with the multi-track music machine, In *Proceedings of the 21st international society for music information retrieval conference, ismir 2020*. https://ddmal.music.mcgill.ca/ISMIR-Conf/static/lbd/ISMIR2020-LBD-423-abstract.pdf

Epps, J., & Holmes, W. H. (1999). A new technique for wideband enhancement of coded narrowband speech, In *1999 IEEE Workshop on Speech Coding Proceedings. Model, Coders, and Error Criteria (Cat. No. 99EX351)*, IEEE. https://ieeexplore.ieee.org/iel5/6345/16956/00781522.pdf

Evans, Z., Carr, C. J., Taylor, J., Hawley, S. H., & Pons, J. (2024). Fast timing-conditioned latent audio diffusion, In *Forty-first international conference on machine learning, icml 2024, vienna, austria, july 21-27, 2024*, OpenReview.net. https://openreview.net/forum?id=jOlO8t1xdx

Falkowski-Gilski, P., & Uhl, T. (2020). Current trends in consumption of multimedia content using online streaming platforms: A user-centric survey. *Computer Science Review*, *37*, 100268. https://doi.org/10.1016/j.cosrev.2020.100268

Fan, Y., Shi, H., Yu, J., Liu, D., Han, W., Yu, H., Wang, Z., Wang, X., & Huang, T. S. (2017). Balanced two-stage residual networks for image super-resolution, In *2017 IEEE conference on computer vision and pattern recognition workshops, CVPR workshops 2017, honolulu, hi, usa, july 21-26, 2017*, IEEE Computer Society. https://doi.org/10.1109/CVPRW.2017.154

Ferreira, L., & Whitehead, J. (2019). Learning to generate music with sentiment, In *Proceedings of the 20th international society for music information retrieval conference, ismir 2019, delft, the netherlands, november 4-8, 2019*. http://archives.ismir.net/ismir2019/paper/000045.pdf

Gan, C., Huang, D., Chen, P., Tenenbaum, J. B., & Torralba, A. (2020). Foley music: Learning to generate music from videos, In *Computer vision - eccv 2020 - 16th european conference*, Springer. https://doi.org/10.1007/978-3-030-58621-8_44

Gao, X., Zhao, Y., Zhang, J., & Cai, L. (2021). Pairwise emotional relationship recognition in drama videos: Dataset and benchmark, In *Proceedings of the 29th acm international conference on multimedia*, Virtual Event China, ACM. https://doi.org/10.1145/3474085.3475493

Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., & Ritter, M. (2017). Audio set: An ontology and human-labeled dataset for audio events, In *2017 ieee international conference on acoustics, speech and signal processing, icassp 2017, new orleans, la, usa, march 5-9, 2017*, IEEE. https://doi.org/10.1109/ICASSP.2017.7952261

Ghodrati, V., Shao, J., Bydder, M., Zhou, Z., Yin, W., Nguyen, K.-L., Yang, Y., & Hu, P. (2019). MR image reconstruction using deep learning: Evaluation of network structure and loss functions. *Quantitative Imaging in Medicine and Surgery*, *9*(9), 1516–1527. https://doi.org/10.21037/qims.2019.08.10

Godsill, S., Rayner, P., & Cappé, O. (2002). Digital audio restoration. In *Applications of digital signal processing to audio and acoustics* (pp. 133–194). Boston, MA, Springer US. https://doi.org/10.1007/0-306-47042-X_4

Gong, Y., Chung, Y.-A., & Glass, J. R. (2021). AST: Audio spectrogram transformer, In *Interspeech 2021, 22nd annual conference of the international speech communication association, brno, czechia, 30 august - 3 september 2021*, ISCA. https://doi.org/10.21437/Interspeech.2021-698

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks, New York, NY, USA, Association for Computing Machinery. https://doi.org/10.1145/3422622

Gunes, H., Schuller, B. W., Pantic, M., & Cowie, R. (2011). Emotion representation, analysis and synthesis in continuous space: A survey, In *Ninth IEEE international conference on automatic face and gesture recognition (FG 2011), santa barbara, ca, usa, 21-25 march 2011*, IEEE Computer Society. https://doi.org/10.1109/FG.2011.5771357

Guo, R., Simpson, I., Magnusson, T., Kiefer, C., & Herremans, D. (2020). A variational autoencoder for music generation controlled by tonal tension, In *Proceedings of the 2020 joint conference on ai music creativity*. https://arxiv.org/abs/2010.06230

Haidt, J., Rozin, P., Mccauley, C., & Imada, S. (1997). Body, psyche, and culture: The relationship between disgust and morality. *Psychology and Developing Societies*, *9*(1), 107–131. https://doi.org/10.1177/097133369700900105

Harper, F. M., & Konstan, J. A. (2016). The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, *5*(4), 19:1–19:19. https://doi.org/10.1145/2827872

Haviv, A., Ram, O., Press, O., Izsak, P., & Levy, O. (2022). Transformer language models without positional encodings still learn positional information, In *Findings of the association for computational linguistics: Emnlp 2022*, Association for Computational Linguistics. https://doi.org/10.18653/V1/2022.FINDINGS-EMNLP.99

Hawthorne, C., Elsen, E., Song, J., Roberts, A., Simon, I., Raffel, C., Engel, J. H., Oore, S., & Eck, D. (2018). Onsets and frames: Dual-objective piano transcription, In *Proceedings of the 19th international society for music information retrieval conference*. http://ismir2018.ircam.fr/doc/pdfs/19%5C_Paper.pdf

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J. H., & Eck, D. (2019). Enabling factorized piano music modeling and generation with the MAESTRO dataset, In *7th international conference on learning representations, ICLR 2019, new orleans, la, usa, may 6-9, 2019*, OpenReview.net. https://openreview.net/forum?id=r1lYRjC9F7

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition, In *Ieee/cvf conference on computer vision and pattern recognition, cvpr 2016*. https://doi.org/10.1109/CVPR.2016.90

Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J., & Wilson, K. W. (2017). CNN architectures for large-scale audio classification, In *2017 IEEE international conference on acoustics, speech and signal processing, ICASSP 2017, new orleans, la, usa, march 5-9, 2017*, IEEE. https://doi.org/10.1109/ICASSP.2017.7952132

Hines, A., Skoglund, J., Kokaram, A. C., & Harte, N. (2012). Visqol: The virtual speech quality objective listener, In *IWAENC 2012 - international workshop on acoustic signal enhancement, proceedings, RWTH aachen university, germany, september 4th - 6th, 2012*, VDE-Verlag. http://www.vde-verlag.de/proceedings-de/453451035.html

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*arXiv. https://doi.org/10.48550/arXiv.1503.02531

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, *9*(8), 1735–1780. https://doi.org/10.1162/NECO.1997.9.8.1735

Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2020). The curious case of neural text degeneration, In *8th international conference on learning representations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*, OpenReview.net. https://openreview.net/forum?id=rygGQyrFvH

Horn, B. K., & Schunck, B. G. (1981). Determining optical flow. *Artificial intelligence*, *17*(1-3), 185–203. https://doi.org/10.1016/0004-3702(81)90024-2

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., & Gelly, S. (2019). Parameter-efficient transfer learning for NLP, In *Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 june 2019, long beach, california, USA*, PMLR. http://proceedings.mlr.press/v97/houlsby19a.html

Hsiao, W.-Y., Liu, J.-Y., Yeh, Y.-C., & Yang, Y.-H. (2021). Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs, In *Proceedings of the aaai conference on artificial intelligence*. https://doi.org/10.1609/AAAI.V35I1.16091

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2022). Lora: Low-rank adaptation of large language models, In *The tenth international conference on learning representations, iclr 2022, virtual event, april 25-29, 2022*, OpenReview.net. Retrieved February 15, 2025, from https://openreview.net/forum?id=nZeVKeeFYf9

Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews, In *Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining, seattle, washington, usa, august 22-25, 2004*, ACM. https://doi.org/10.1145/1014052.1014073

Huang, C.-Z. A., Cooijmans, T., Roberts, A., Courville, A. C., & Eck, D. (2017). Counterpoint by convolution, In *Proceedings of the 18th international society for music information retrieval conference, ISMIR 2017, suzhou, china, october 23-27, 2017*. https://ismir2017.smcnus.org/wp-content/uploads/2017/10/187%5C_Paper.pdf

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Simon, I., Hawthorne, C., Shazeer, N., Dai, A. M., Hoffman, M. D., Dinculescu, M., & Eck, D. (2019). Music transformer: Generating music with long-term structure, In *7th international conference on learning representations, ICLR 2019, new orleans, la, usa, may 6-9, 2019*, OpenReview.net. https://openreview.net/forum?id=rJe4ShAcF7

Huang, Q., Xiong, Y., Rao, A., Wang, J., & Lin, D. (2020). MovieNet: A holistic dataset for movie understanding, In *Computer vision - eccv 2020 - 16th european conference, glasgow, uk, august 23-28, 2020, proceedings, part iv*, Springer. https://doi.org/10.1007/978-3-030-58548-8_41

Hung, H.-T., Ching, J., Doh, S., Kim, N., Nam, J., & Yang, Y.-H. (2021). EMOPIA: A multimodal pop piano dataset for emotion recognition and emotion-based music generation, In *Proceedings of the 22nd international society for music information retrieval conference, ISMIR 2021, online, november 7-12, 2021*. https://archives.ismir.net/ismir2021/paper/000039.pdf

Hunter, P. G., & Schellenberg, E. G. (2010). Music and emotion. In *Music perception* (pp. 129–164). New York, NY, Springer New York. https://doi.org/10.1007/978-1-4419-6114-3_5

Hussain, Z., Zhang, M., Zhang, X., Ye, K., Thomas, C., Agha, Z., Ong, N., & Kovashka, A. (2017). Automatic understanding of image and video advertisements, In *2017 IEEE conference on computer vision and pattern recognition, CVPR 2017, honolulu, hi, usa, july 21-26, 2017*, IEEE Computer Society. https://doi.org/10.1109/CVPR.2017.123

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift, In *Proceedings of the 32nd international conference on machine learning, ICML 2015, lille, france, 6-11 july 2015*, JMLR.org. http://proceedings.mlr.press/v37/ioffe15.html

Iqbal, T., & Qureshi, S. (2022). The survey: Text generation models in deep learning. *Journal of King Saud University - Computer and Information Sciences*, *34*(6, Part A), 2515–2528. https://doi.org/10.1016/j.jksuci.2020.04.001

Iser, B., & Schmidt, G. (2003). Neural networks versus codebooks in an application for bandwidth extension of speech signals, In *8th european conference on speech communication and technology, EUROSPEECH 2003 - INTERSPEECH 2003, geneva, switzerland, september 1-4, 2003*, ISCA. https://doi.org/10.21437/EUROSPEECH.2003-227

Iyyer, M., Enns, P., Boyd-Graber, J. L., & Resnik, P. (2014). Political ideology detection using recursive neural networks, In *Proceedings of the 52nd annual meeting of the association for computational linguistics, ACL 2014, june 22-27, 2014, baltimore, md, usa, volume 1: Long papers*, The Association for Computer Linguistics. https://doi.org/10.3115/V1/P14-1105

Izard, C. E. (2013). *Human emotions*. Springer Science & Business Media. http://link.springer.com/10.1007/978-1-4899-2209-0

Jansson, A., Humphrey, E. J., Montecchio, N., Bittner, R. M., Kumar, A., & Weyde, T. (2017). Singing voice separation with deep u-net convolutional networks, In *Proceedings of the 18th international society for music information retrieval conference, ISMIR 2017,*

suzhou, china, october 23-27, 2017. https://ismir2017.smcnus.org/wp-content/uploads/2017/10/171%5C_Paper.pdf

Jax, P., & Vary, P. (2003). On artificial bandwidth extension of telephone speech. *Signal Processing*, *83*(8), 1707–1719. https://doi.org/10.1016/S0165-1684(03)00082-3

Jhuang, H., Gall, J., Zuffi, S., Schmid, C., & Black, M. J. (2013). Towards understanding action recognition, In *Proceedings of the ieee international conference on computer vision*. Retrieved February 20, 2025, from http://openaccess.thecvf.com/content_iccv_2013/html/Jhuang_Towards_Understanding_Action_2013_ICCV_paper.html

Jiang, Y.-G., Xu, B., & Xue, X. (2014). Predicting emotions in user-generated videos. *Proceedings of the AAAI Conference on Artificial Intelligence*, *28*(1). https://doi.org/10.1609/aaai.v28i1.8724

Johnson, D. D. (2017). Generating polyphonic music using tied parallel networks, In *Computational intelligence in music, sound, art and design*, Amsterdam, The Netherlands, Springer. https://doi.org/10.1007/978-3-319-55750-2_9

Juslin, P. N., & Sloboda, J. A. (2001). *Music and emotion: Theory and research*. Oxford University Press. https://doi.org/10.1093/oso/9780192631886.001.0001

Kang, J., Poria, S., & Herremans, D. (2024). Video2music: Suitable music generation from videos using an affective multimodal transformer model. *Expert Systems with Applications*, *249*, 123640. https://doi.org/10.1016/j.eswa.2024.123640

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling laws for neural language models. *CoRR*, *abs/2001.08361*arXiv 2001.08361. https://arxiv.org/abs/2001.08361

Kazemnejad, A., Padhi, I., Natesan Ramamurthy, K., Das, P., & Reddy, S. (2024). The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, *36*. https://proceedings.neurips.cc/paper_files/paper/2023/hash/4e85362c02172c0c6567ce593122d31c-Abstract-Conference.html

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., & Socher, R. (2019). CTRL: A conditional transformer language model for controllable generation. *CoRR*, *abs/1909.05858*arXiv 1909.05858. http://arxiv.org/abs/1909.05858

Keung, P., Lu, Y., Szarvas, G., & Smith, N. A. (2020). The multilingual amazon reviews corpus, In *Proceedings of the 2020 conference on empirical methods in natural language processing, EMNLP 2020, online, november 16-20, 2020*, Association for Computational Linguistics. https://doi.org/10.18653/V1/2020.EMNLP-MAIN.369

Khan, H., Hussain, T., Ullah Khan, S., Ahmad Khan, Z., & Baik, S. W. (2024). Deep multi-scale pyramidal features network for supervised video summarization. *Expert Systems with Applications*, *237*, 121288. https://doi.org/10.1016/j.eswa.2023.121288

Kilgour, K., Zuluaga, M., Roblek, D., & Sharifi, M. (2019). Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms (G. Kubin & Z. Kacic, Eds.). In G. Kubin & Z. Kacic (Eds.), *20th annual conference of the international speech communication association, interspeech 2019, graz, austria, september 15-19, 2019*, ISCA. https://doi.org/10.21437/INTERSPEECH.2019-2219

Kim, J., Lee, S., Kim, S., & Yoo, W. Y. (2011). Music mood classification model based on arousal-valence values, In *13th international conference on advanced communication technology (icact2011)*, IEEE. https://ieeexplore.ieee.org/iel5/5740523/5745722/05745796.pdf

Kim, S., & Sathe, V. (2019). Bandwidth extension on raw audio via generative adversarial networks. *CoRR*, *abs/1903.09027*arXiv 1903.09027. http://arxiv.org/abs/1903.09027

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization, In *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings*. http://arxiv.org/abs/1412.6980

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks, In *5th international conference on learning representations, iclr 2017*. https://openreview.net/forum?id=SJU4ayYgl

Koelsch, S. (2014). Brain correlates of music-evoked emotions. *Nature Reviews Neuroscience*, *15*(3), 170–180. https://doi.org/10.1038/nrn3666

Koepke, A. S., Wiles, O., Moses, Y., & Zisserman, A. (2020). Sight to sound: An end-to-end approach for visual piano transcription, In *Ieee international conference on acoustics, speech and signal processing, icassp 2020*, IEEE. https://doi.org/10.1109/ICASSP40776.2020.9053115

Kong, Q., Cao, Y., Iqbal, T., Wang, Y., Wang, W., & Plumbley, M. D. (2020). PANNs: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE ACM Transactions on Audio, Speech, and Language Processing*, *28*, 2880–2894. https://doi.org/10.1109/TASLP.2020.3030497

Kontio, J., Laaksonen, L., & Alku, P. (2007). Neural Network-Based Artificial Bandwidth Expansion of Speech. *IEEE Transactions on Audio, Speech and Language Processing*, *15*(3), 873–881. https://doi.org/10.1109/TASL.2006.885934

Krishna, K., Wieting, J., & Iyyer, M. (2020). Reformulating unsupervised style transfer as paraphrase generation, In *Proceedings of the 2020 conference on empirical methods in natural language processing, EMNLP 2020, online, november 16-20, 2020*, Association for Computational Linguistics. https://doi.org/10.18653/V1/2020.EMNLP-MAIN.55

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks, In *Advances in neural information processing systems 25: 26th annual conference on neural information processing systems 2012. proceedings of a meeting held december 3-6, 2012, lake tahoe, nevada, united states*. https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

Krumhansl, C. L. (2002). Music: A link between cognition and emotion. *Current Directions in Psychological Science*, *11*(2), https://doi.org/10.1111/1467-8721.00165, 45–50. https://doi.org/10.1111/1467-8721.00165

Kuleshov, V. (2020). Kuleshov/audio-super-res. https://github.com/kuleshov/audio-super-res

Kuleshov, V., Enam, S. Z., & Ermon, S. (2017). Audio super-resolution using neural networks, In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, workshop track proceedings*, OpenReview.net. https://openreview.net/forum?id=S1gNakBFx

Kusal, S., Patil, S., Choudrie, J., Kotecha, K., Vora, D. R., & Pappas, I. O. (2022). A review on text-based emotion detection - techniques, applications, datasets, and future directions. *CoRR*, *abs/2205.03235*arXiv 2205.03235. https://doi.org/10.48550/ARXIV.2205.03235

Lam, M. W. Y., Tian, Q., Li, T., Yin, Z., Feng, S., Tu, M., Ji, Y., Xia, R., Ma, M., Song, X., Chen, J., Wang, Y., & Wang, Y. (2023). Efficient neural music generation, In *Advances in neural information processing systems 36: Annual conference on neural information processing systems 2023, neurips 2023, new orleans, la, usa, december 10 - 16, 2023*. http://papers.nips.cc/paper%5C_files/paper/2023/hash/38b23e2328096520e9c889ae03e372c9-Abstract-Conference.html

Lee, J., Kim, S., Kim, S., Park, J., & Sohn, K. (2019). Context-aware emotion recognition networks, In *2019 ieee/cvf international conference on computer vision, iccv 2019, seoul,*

korea (south), october 27 - november 2, 2019, IEEE. https://doi.org/10.1109/ICCV.2019.01024

Leon, F. L. D., Wang, Y., Feng, Y., & Lee, M. G. (2025). Uob-nlp at semeval-2025 task 11: Leveraging adapters for multilingual and cross-lingual emotion detection. arXiv. https://doi.org/10.48550/arXiv.2504.08543

Levesque, H. J., Davis, E., & Morgenstern, L. (2012). The winograd schema challenge, In *Principles of knowledge representation and reasoning: Proceedings of the thirteenth international conference, KR 2012, rome, italy, june 10-14, 2012*, AAAI Press. http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4492

Li, B., Liu, X., Dinesh, K., Duan, Z., & Sharma, G. (2019). Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications. *IEEE Transactions on Multimedia*, *21*(2), 522–535. https://doi.org/10.1109/TMM.2018.2856090

Li, K., Huang, Z., Xu, Y., & Lee, C.-H. (2015). Dnn-based speech bandwidth expansion and its application to adding high-frequency missing features for automatic speech recognition of narrowband speech, In *16th annual conference of the international speech communication association, INTERSPEECH 2015, dresden, germany, september 6-10, 2015*, ISCA. https://doi.org/10.21437/INTERSPEECH.2015-555

Li, Y., Tagliasacchi, M., Gfeller, B., & Roblek, D. (2020). Learning to denoise historical music, In *Proceedings of the 21th international society for music information retrieval conference, ISMIR 2020, montreal, canada, october 11-16, 2020*. http://archives.ismir.net/ismir2020/paper/000057.pdf

Lim, B., Son, S., Kim, H., Nah, S., & Lee, K. M. (2017). Enhanced deep residual networks for single image super-resolution, In *2017 IEEE conference on computer vision and pattern recognition workshops, CVPR workshops 2017, honolulu, hi, usa, july 21-26, 2017*, IEEE Computer Society. https://doi.org/10.1109/CVPRW.2017.151

Lim, T.-Y., Yeh, R. A., Xu, Y., Do, M. N., & Hasegawa-Johnson, M. (2018). Time-frequency networks for audio super-resolution, In *2018 IEEE international conference on acoustics, speech and signal processing, ICASSP 2018, calgary, ab, canada, april 15-20, 2018*, IEEE. https://doi.org/10.1109/ICASSP.2018.8462049

Liu, B., & Zhang, L. (2012). A survey of opinion mining and sentiment analysis. In *Mining text data* (pp. 415–463). Springer. https://doi.org/10.1007/978-1-4614-3223-4_13

Liu, H., Yuan, Y., Liu, X., Mei, X., Kong, Q., Tian, Q., Wang, Y., Wang, W., Wang, Y., & Plumbley, M. D. (2024). Audioldm 2: Learning holistic audio generation with self-supervised pretraining. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *32*, 2871–2883. https://doi.org/10.1109/TASLP.2024.3399607

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv. https://doi.org/10.48550/arXiv.1907.11692

Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., & Hu, H. (2022). Video swin transformer, In *Ieee/cvf conference on computer vision and pattern recognition, cvpr 2022*, IEEE. https://doi.org/10.1109/CVPR52688.2022.00320

Liutkus, A., Stöter, F.-R., Rafii, Z., Kitamura, D., Rivet, B., Ito, N., Ono, N., & Fontecave, J. (2017). The 2016 signal separation evaluation campaign, In *Latent variable analysis and signal separation - 13th international conference, LVA/ICA 2017, grenoble, france, february 21-23, 2017, proceedings*. https://doi.org/10.1007/978-3-319-53547-0_31

Macartney, C., & Weyde, T. (2018). Improved speech enhancement with the wave-u-net. *CoRR*, *abs/1811.11307*arXiv 1811.11307. http://arxiv.org/abs/1811.11307

Manocha, P., Finkelstein, A., Zhang, R., Bryan, N. J., Mysore, G. J., & Jin, Z. (2020). A differentiable perceptual audio metric learned from just noticeable differences, In *21st annual conference of the international speech communication association, interspeech 2020, virtual event, shanghai, china, october 25-29, 2020*, ISCA. https://doi.org/10.21437/INTERSPEECH.2020-1191

Matsuda, Y.-T., Fujimura, T., Katahira, K., Okada, M., Ueno, K., Cheng, K., & Okanoya, K. (2013). The implicit processing of categorical and dimensional strategies: An fmri study of facial emotion perception. *Frontiers in Human Neuroscience*, *Volume 7 - 2013*. https://doi.org/10.3389/fnhum.2013.00551

May, P. (2021). Machine translated multilingual sts benchmark dataset. https://github.com/PhilipMay/stsb-multi-mt

McFee, B., Humphrey, E. J., & Bello, J. P. (2015). A software framework for musical data augmentation, In *Proceedings of the 16th international society for music information retrieval conference, ISMIR 2015, málaga, spain, october 26-30, 2015*. http://ismir2015.uma.es/articles/228%5C_Paper.pdf

Meyer, L. B. (2008). *Emotion and meaning in music*. University of Chicago Press. https://press.uchicago.edu/ucp/books/book/chicago/E/bo28551887.html

Mikolov, T., & Zweig, G. (2012). Context dependent recurrent neural network language model, In *2012 IEEE spoken language technology workshop (slt), miami, fl, usa, december 2-5, 2012*, IEEE. https://doi.org/10.1109/SLT.2012.6424228

Miranda, E. R., Magee, W. L., Wilson, J. J., Eaton, J., & Palaniappan, R. (2011). Brain-computer music interfacing (bcmi): From basic research to the real world of special needs. *Music & Medicine*, *3*(3), 134–140. https://doi.org/10.47513/mmd.v3i3.370

Miron, M., & Davies, M. E. P. (2018). High frequency magnitude spectrogram reconstruction for music mixtures using convolutional autoencoders, In *21st international conference on digital audio effects (dafx2018)*. https://www.researchgate.net/publication/329372321_High_frequency_magnitude_spectrogram_reconstruction_for_music_mixtures_using_convolutional_autoencoders

Miyazawa, K., Kyuragi, Y., & Nagai, T. (2022). Simple and effective multimodal learning based on pre-trained transformer models. *IEEE Access*, *10*, 29821–29833. https://doi.org/10.1109/ACCESS.2022.3159346

Mohammad, S. M., Bravo-Marquez, F., Salameh, M., & Kiritchenko, S. (2018). Semeval-2018 task 1: Affect in tweets, In *Proceedings of the 12th international workshop on semantic evaluation, semeval@naacl-hlt 2018, new orleans, louisiana, usa, june 5-6, 2018*, Association for Computational Linguistics. https://doi.org/10.18653/V1/S18-1001

Mokady, R., Hertz, A., & Bermano, A. H. (2021). Clipcap: CLIP prefix for image captioning. *CoRR*, *abs/2111.09734*arXiv 2111.09734. https://arxiv.org/abs/2111.09734

Morgan, S. D. (2019). Categorical and dimensional ratings of emotional speech: Behavioral findings from the morgan emotional speech set. *Journal of Speech, Language, and Hearing Research*, *62*(11), 4015–4029. https://doi.org/10.1044/2019_JSLHR-S-19-0144

Muhammad, S. H., Ousidhoum, N., Abdulmumin, I., Yimam, S. M., Wahle, J. P., Ruas, T., Beloucif, M., de Kock, C., Belay, T. D., Ahmad, I. S., Surange, N., Teodorescu, D., Adelani, D. I., Aji, A. F., Ali, F., Araujo, V., Ayele, A. A., Ignat, O., Panchenko, A., … Mohammad, S. M. (2025). Semeval-2025 task 11: Bridging the gap in text-based emotion detection. *CoRR*, *abs/2503.07269*arXiv 2503.07269. https://doi.org/10.48550/ARXIV.2503.07269

Müller, M. (2015). *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Cham, Springer International Publishing. https://doi.org/10.1007/978-3-319-21945-5

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines, In *Proceedings of the 27th international conference on machine learning (icml-10)*. Retrieved February 22, 2025, from https://www.cs.toronto.edu/~hinton/absps/reluICML.pdf

Nakatoh, Y., Tsushima, M., & Norimatsu, T. (1997). Generation of broadband speech from narrowband speech using piecewise linear mapping, In *Fifth european conference on speech communication and technology, EUROSPEECH 1997, rhodes, greece, september 22-25, 1997*, ISCA. https://doi.org/10.21437/EUROSPEECH.1997-469

Nguyen, T. H., Shirai, K., & Velcin, J. (2015). Sentiment analysis on social media for stock movement prediction. *Expert Systems with Applications*, *42*(24), 9603–9611. https://doi.org/10.1016/J.ESWA.2015.07.052

Nitanda, N., Haseyama, M., & Kitajima, H. (2005). Audio signal segmentation and classification for scene-cut detection, In *International symposium on circuits and systems (ISCAS 2005), 23-26 may 2005, kobe, japan*, IEEE. https://doi.org/10.1109/ISCAS.2005.1465515

Niu, S., Liu, Y., Wang, J., & Song, H. (2020). A decade survey of transfer learning (2010–2020). *IEEE Transactions on Artificial Intelligence*, *1*(2), 151–166. https://doi.org/10.1109/TAI.2021.3054609

Nour-Eldin, A. H., & Kabal, P. (2008). Mel-frequency cepstral coefficient-based bandwidth extension of narrowband speech, In *9th annual conference of the international speech communication association, INTERSPEECH 2008, brisbane, australia, september 22-26, 2008*, ISCA. https://doi.org/10.21437/INTERSPEECH.2008-11

Ojala, T., Pietikäinen, M., & Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, *29*(1), 51–59. https://doi.org/10.1016/0031-3203(95)00067-4

Ojha, A. K., Doğruöz, A. S., Madabushi, H. T., Martino, G. D. S., Rosenthal, S., & Rosá, A. (Eds.). (2024). *Proceedings of the 18th international workshop on semantic evaluation, semeval@naacl 2024, mexico city, mexico, june 20-21, 2024*, Association for Computational Linguistics. https://aclanthology.org/volumes/2024.semeval-1/

Oore, S., Simon, I., Dieleman, S., Eck, D., & Simonyan, K. (2020). This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, *32*(4), 955–967. https://doi.org/10.1007/S00521-018-3758-9

Ovalle, J. E. A., Solorio, T., Montes-y-Gómez, M., & González, F. A. (2017). Gated multimodal units for information fusion, In *5th international conference on learning representations, iclr 2017, toulon, france, april 24-26, 2017, workshop track proceedings*, OpenReview.net. Retrieved February 24, 2025, from https://openreview.net/forum?id=S12%5C_nquOe

Pan, J., Wang, S., & Fang, L. (2022). Representation learning through multimodal attention and time-sync comments for affective video content analysis, In *MM '22: The 30th ACM international conference on multimedia, lisboa, portugal, october 10 - 14, 2022*, ACM. https://doi.org/10.1145/3503161.3548018

Panda, R., Malheiro, R., Rocha, B., Oliveira, A., & Paiva, R. P. (2013). Multi-modal Music Emotion Recognition: A New Dataset, Methodology and Comparative Analysis, In *International Symposium on Computer Music Multidisciplinary Research*. https://hdl.handle.net/10316/94095

Pandeya, Y. R., & Lee, J. (2021). Deep learning-based late fusion of multimodal information for emotion classification of music video. *Multimedia Tools and Applications*, *80*(2), 2887–2905. https://doi.org/10.1007/s11042-020-08836-3

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques, In *Proceedings of the 2002 conference on empirical methods in natural language processing, EMNLP 2002, philadelphia, pa, usa, july 6-7, 2002*. https://doi.org/10.3115/1118693.1118704

Park, K.-Y., & Kim, H. S. (2000a). Narrowband to wideband conversion of speech using GMM based transformation, In *IEEE international conference on acoustics, speech, and signal processing. ICASSP 2000, 5-9 june, 2000, hilton hotel and convention center, istanbul, turkey*, IEEE. https://doi.org/10.1109/ICASSP.2000.862114

Park, K.-Y., & Kim, H. S. (2000b). Narrowband to wideband conversion of speech using GMM based transformation, In *IEEE international conference on acoustics, speech, and signal processing. ICASSP 2000, 5-9 june, 2000, hilton hotel and convention center, istanbul, turkey*, IEEE. https://doi.org/10.1109/ICASSP.2000.862114

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library, In *Advances in neural information processing systems 32: Annual conference on neural information processing systems 2019, neurips 2019, 8-14 december 2019, vancouver, bc, canada*. https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

Payne, C. (2019). Musenet. *OpenAI Blog*, *3*. https://openai.com/blog/musenet

Perraudin, N., Holighaus, N., Majdak, P., & Balazs, P. (2018). Inpainting of long audio segments with similarity graphs. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *26*(6), 1079–1090. https://doi.org/10.1109/TASLP.2018.2809864

Pimpale, M. M. R., Therese, S., & Shinde, V. (2016). A survey on: Sound source separation methods. *International Journal*, *3*(11), 580–584. https://www.academia.edu/download/50739135/V3I1103.pdf

Plutchik, R. (1988). The nature of emotions: Clinical implications. In *Emotions and psychopathology* (pp. 1–20). Boston, MA, Springer US. https://doi.org/10.1007/978-1-4757-1987-1_1

Poria, S., Hazarika, D., Majumder, N., Naik, G., Cambria, E., & Mihalcea, R. (2019). MELD: A multimodal multi-party dataset for emotion recognition in conversations, In *Proceedings of the 57th conference of the association for computational linguistics, ACL 2019, florence, italy, july 28- august 2, 2019, volume 1: Long papers*, Association for Computational Linguistics. https://doi.org/10.18653/V1/P19-1050

Pratt, R. R., Abel, H.-H., & Skidmore, J. (1995). The effects of neurofeedback training with background music on eeg patterns of add and adhd children. *International Journal of Arts Medicine*, *4*(1), 24–31. https://psycnet.apa.org/record/1996-02780-004

Prendergast, R. M. (1992). *Film music: A neglected art: A critical study of music in films*. WW Norton & Company. https://books.google.com/books/about/Film_Music.html?id=7ozTus2HZbsC

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision, In *Proceedings of the 38th international conference on machine learning, ICML 2021, 18-24 july 2021, virtual event*, PMLR. http://proceedings.mlr.press/v139/radford21a.html

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023). Robust speech recognition via large-scale weak supervision, In *International conference on machine learning, icml 2023, 23-29 july 2023, honolulu, hawaii, usa*, PMLR. https://proceedings.mlr.press/v202/radford23a.html

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving language understanding by generative pre-training*. OpenAI. http://www.nlpir.org/wordpress/wp-content/uploads/2019/06/Improving-language-understanding-by-generative-pre-training.pdf

Raffel, C. (2016). *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching* (Doctoral dissertation). Columbia University, USA. https://doi.org/10.7916/D8N58MHV

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*(1).

Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I., & Bittner, R. (2017). The MUSDB18 corpus for music separation. https://inria.hal.science/hal-02190845/document

Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I., & Bittner, R. (2019). MUSDB18-HQ - an uncompressed version of MUSDB18. Zenodo. https://doi.org/10.5281/zenodo.3338372

Ranjbar, N., & Baghbani, H. (2025). Lotus at semeval-2025 task 11: Roberta with llama-3 generated explanations for multi-label emotion classification. arXiv. https://doi.org/10.48550/arXiv.2502.19935

Ray, A., & Kolekar, M. H. (2024). Transfer learning and its extensive appositeness in human activity recognition: A survey. *Expert Systems with Applications*, *240*, 122538. https://doi.org/10.1016/j.eswa.2023.122538

Redmon, J., Divvala, S. K., Girshick, R. B., & Farhadi, A. (2016). You only look once: Unified, real-time object detection, In *2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, las vegas, nv, usa, june 27-30, 2016*, IEEE Computer Society. https://doi.org/10.1109/CVPR.2016.91

Reiss, T., Cohen, N., Bergman, L., & Hoshen, Y. (2021). PANDA: adapting pretrained features for anomaly detection and segmentation, In *IEEE conference on computer vision and pattern recognition, CVPR 2021, virtual, june 19-25, 2021*, Computer Vision Foundation / IEEE. https://doi.org/10.1109/CVPR46437.2021.00283

Rempfler, M., Kumar, S., Stierle, V., Paulitschke, P., Andres, B., & Menze, B. H. (2017). Cell lineage tracing in lens-free microscopy videos, In *Medical image computing and computer assisted intervention - MICCAI 2017 - 20th international conference, quebec city, qc, canada, september 11-13, 2017, proceedings, part II*, Springer. https://doi.org/10.1007/978-3-319-66185-8_1

Rix, A. W., Beerends, J. G., Hollier, M. P., & Hekstra, A. P. (2001). Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs, In *IEEE international conference on acoustics, speech, and signal processing, ICASSP 2001, 7-11 may, 2001, salt palace convention center, salt lake city, utah, usa, proceedings*, IEEE. https://doi.org/10.1109/ICASSP.2001.941023

Roberts, A., Engel, J. H., Raffel, C., Hawthorne, C., & Eck, D. (2018). A hierarchical latent vector model for learning long-term structure in music, In *Proceedings of the 35th international conference on machine learning, ICML 2018, stockholmsmässan, stockholm, sweden, july 10-15, 2018*, PMLR. http://proceedings.mlr.press/v80/roberts18a.html

Rodriguez Bribiesca, I., Lopez Monroy, A. P., & Montes-y-Gomez, M. (2021). Multimodal weighted fusion of transformers for movie genre classification, In *Proceedings of the third workshop on multimodal artificial intelligence*, Mexico City, Mexico, Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.maiworkshop-1.1

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation, In *Medical image computing and computer-assisted intervention -*

MICCAI 2015 - 18th international conference munich, germany, october 5 - 9, 2015, proceedings, part III, Springer. https://doi.org/10.1007/978-3-319-24574-4_28

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Russell, J. A. (1980). A circumplex model of affect. *Journal of personality and social psychology*, *39*(6), 1161. https://doi.org/10.1037/h0077714

Russell, J. A., & Mehrabian, A. (1977). Evidence for a three-factor theory of emotions. *Journal of Research in Personality*, *11*(3), 273–294. https://doi.org/https://doi.org/10.1016/0092-6566(77)90037-X

Sambaragi, L. M., Pradeep Naik, P., Kakatkar, N. N., Chikkamath, S., S R, N., & Budihal, S. V. (2024). Music generation: A simplified approach, In *2024 3rd international conference for innovation in technology (inocon)*. https://doi.org/10.1109/INOCON60754.2024.10511477

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*. https://doi.org/10.48550/arXiv.1910.01108

Sautter, J., Faubel, F., Buck, M., & Schmidt, G. (2019). Artificial bandwidth extension using a conditional generative adversarial network with discriminative training, In *IEEE international conference on acoustics, speech and signal processing, ICASSP 2019, brighton, united kingdom, may 12-17, 2019*, IEEE. https://doi.org/10.1109/ICASSP.2019.8682649

Savchenko, A. V. (2023). Emotieffnets for facial processing in video-based valence-arousal prediction, expression classification and action unit detection, In *IEEE/CVF conference on computer vision and pattern recognition, CVPR 2023 - workshops, vancouver, bc, canada, june 17-24, 2023*, IEEE. https://doi.org/10.1109/CVPRW59228.2023.00606

Schifrin, L. (2011). *Music composition for film and television*. Hal Leonard Corporation. https://books.google.com/books/about/Music_Composition_for_Film_and_Televisio.html?id=gTwLAQAAQBAJ

Sennrich, R., Haddow, B., & Birch, A. (2016a). Controlling politeness in neural machine translation via side constraints, In *Naacl hlt 2016, the 2016 conference of the north american chapter of the association for computational linguistics: Human language technologies, san diego california, usa, june 12-17, 2016*. https://doi.org/10.18653/v1/n16-1005

Sennrich, R., Haddow, B., & Birch, A. (2016b). Neural machine translation of rare words with subword units, In *Proceedings of the 54th annual meeting of the association for computational linguistics, acl 2016, august 7-12, 2016, berlin, germany, volume 1: Long papers*, The Association for Computer Linguistics. https://doi.org/10.18653/V1/P16-1162

Sevilla, J., Heim, L., Ho, A., Besiroglu, T., Hobbhahn, M., & Villalobos, P. (2022). Compute trends across three eras of machine learning, In *International joint conference on neural networks, IJCNN 2022, padua, italy, july 18-23, 2022*, IEEE. https://doi.org/10.1109/IJCNN55064.2022.9891914

Shahraki, A. G. (2015). *Emotion mining from text* (Master's thesis). University of Alberta. https://era.library.ualberta.ca/items/27ae961f-d9a6-4a5a-9b6f-f180478ea573

Shaw, P., Uszkoreit, J., & Vaswani, A. (2018). Self-attention with relative position representations, In *Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, naacl-hlt, new orleans, louisiana, usa, june 1-6, 2018, volume 2 (short papers)*, Association for Computational Linguistics. https://doi.org/10.18653/V1/N18-2074

Sheng, E., Chang, K.-W., Natarajan, P., & Peng, N. (2020). Towards controllable biases in language generation, In *Proceedings of the 2020 conference on empirical methods in natural*

language processing: Findings, emnlp 2020, online event, 16-20 november 2020. https://doi.org/10.18653/v1/2020.findings-emnlp.291

Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., & Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, In *2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, las vegas, nv, usa, june 27-30, 2016*, IEEE Computer Society. https://doi.org/10.1109/CVPR.2016.207

Simoes, G. S., Wehrmann, J., Barros, R. C., & Ruiz, D. D. (2016). Movie genre classification with convolutional neural networks, In *2016 international joint conference on neural networks, IJCNN 2016, vancouver, bc, canada, july 24-29, 2016*, IEEE. https://doi.org/10.1109/IJCNN.2016.7727207

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. http://arxiv.org/abs/1409.1556

Smith, E. M., Gonzalez-Rico, D., Dinan, E., & Boureau, Y.-L. (2020). Controlling style in generated dialogue. *CoRR*, *abs/2009.10855*arXiv 2009.10855. https://arxiv.org/abs/2009.10855

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics, In *Proceedings of the 32nd international conference on machine learning, ICML 2015, lille, france, 6-11 july 2015*, JMLR.org. http://proceedings.mlr.press/v37/sohl-dickstein15.html

Soleymani, M., Pantic, M., & Pun, T. (2012). Multimodal emotion recognition in response to videos. *IEEE Transactions on Affective Computing*, *3*(2), 211–223. https://doi.org/10.1109/T-AFFC.2011.37

Song, G.-B., & Martynovich, P. (2009). A study of HMM-based bandwidth extension of speech signals. *Signal Processing*, *89*(10), 2036–2044. https://doi.org/10.1016/j.sigpro.2009.03.037

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958. https://doi.org/10.5555/2627435.2670313

Stratton, V. N., & Zalanowski, A. H. (1994). Affective impact of music vs. lyrics. *Empirical Studies of the Arts*, *12*(2), 173–184. https://doi.org/10.2190/35T0-U4DT-N09Q-LQHW

Su, K., Liu, X., & Shlizerman, E. (2020). Audeo: Audio generation for a silent performance video, In *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, neurips 2020, december 6-12, 2020, virtual*. https://proceedings.neurips.cc/paper/2020/hash/227f6afd3b7f89b96c4bb91f95d50f6d-Abstract.html

Su, K., Liu, X., & Shlizerman, E. (2021). How does it sound?, In *Advances in neural information processing systems 34: Annual conference on neural information processing systems 2021, neurips 2021, december 6-14, 2021, virtual*. https://proceedings.neurips.cc/paper/2021/hash/f4e369c0a468d3aeeda0593ba90b5e55-Abstract.html

Sulun, S. (2018). Deep learned frame prediction for video compression. *arXiv preprint arXiv:1811.10946*arXiv 1811.10946. http://arxiv.org/abs/1811.10946

Sulun, S., & Davies, M. E. P. (2021). On filter generalization for music bandwidth extension using deep neural networks. *IEEE Journal of Selected Topics in Signal Processing*, *15*(1), 132–142. https://doi.org/10.1109/JSTSP.2020.3037485

Sulun, S., Davies, M. E. P., & Viana, P. (2022). Symbolic music generation conditioned on continuous-valued emotions. *IEEE Access*, *10*, 44617–44626. https://doi.org/10.1109/ACCESS.2022.3169744

Sulun, S., Oliveira, P., & Viana, P. (2023). Emotion4MIDI: A lyrics-based emotion-labeled symbolic music dataset, In *Progress in artificial intelligence - 22nd EPIA conference on artificial intelligence, EPIA 2023, faial island, azores, september 5-8, 2023, proceedings, part II*, Springer. https://doi.org/10.1007/978-3-031-49011-8_7

Sulun, S., & Tekalp, A. M. (2021). Can learned frame prediction compete with block motion compensation for video coding? *Signal Image Video Process.*, *15*(2), 401–410. https://doi.org/10.1007/S11760-020-01751-Y

Sulun, S., Viana, P., & Davies, M. E. P. (2024a). Movie trailer genre classification using multimodal pretrained features. *Expert Systems with Applications*, 125209. https://doi.org/10.1016/j.eswa.2024.125209

Sulun, S., Viana, P., & Davies, M. E. P. (2024b). VEMOCLAP: A video emotion classification web application, In *2024 international symposium on multimedia (ism)*, IEEE Computer Society. https://doi.org/10.1109/ISM63611.2024.00029

Sulun, S., Viana, P., & Davies, M. E. P. (2025). Video soundtrack generation by aligning emotions and temporal boundaries. *CoRR*, *abs/2502.10154*arXiv 2502.10154. https://doi.org/10.48550/ARXIV.2502.10154

Sun, D. L., & Mazumder, R. (2013). Non-negative matrix completion for bandwidth extension: A convex optimization approach, In *IEEE international workshop on machine learning for signal processing, MLSP 2013, southampton, united kingdom, september 22-25, 2013*, IEEE. https://doi.org/10.1109/MLSP.2013.6661924

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018). A survey on deep transfer learning. In *Artificial neural networks and machine learning – icann 2018* (pp. 270–279). Cham, Springer International Publishing. https://doi.org/10.1007/978-3-030-01424-7_27

Tan, H. H., & Herremans, D. (2020). Music fadernets: Controllable music generation based on high-level features via low-level feature modelling, In *Proceedings of the 21th international society for music information retrieval conference, ISMIR 2020, montreal, canada, october 11-16, 2020*. http://archives.ismir.net/ismir2020/paper/000222.pdf

Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks, In *Proceedings of the 36th international conference on machine learning*, PMLR. https://proceedings.mlr.press/v97/tan19a.html

Taylor, L., & Nitschke, G. (2018). Improving deep learning with generic data augmentation, In *IEEE symposium series on computational intelligence, SSCI 2018, bangalore, india, november 18-21, 2018*, IEEE. https://doi.org/10.1109/SSCI.2018.8628742

Torcoli, M., Wu, C.-W., Dick, S., Williams, P. A., Modar Halimeh, M., Wolcott, W., & Habets, E. A. P. (2024). Odaq: Open dataset of audio quality, In *Icassp 2024 - 2024 ieee international conference on acoustics, speech and signal processing (icassp)*. https://doi.org/10.1109/ICASSP48485.2024.10447634

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need, In *Advances in neural information processing systems 30: Annual conference on neural information processing systems 2017, december 4-9, 2017, long beach, ca, USA*. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2017). Show and tell: Lessons learned from the 2015 MSCOCO image captioning challenge. *IEEE Transactions on Pattern Analysis Machine Intelligence*, *39*(4), 652–663. https://doi.org/10.1109/TPAMI.2016.2587640

Viveros-Muñoz, R., Huijse, P., Vargas, V., Espejo, D., Poblete, V., Arenas, J. P., Vernier, M., Vergara, D., & Suárez, E. (2023). Dataset for polyphonic sound event detection tasks in urban soundscapes: The synthetic polyphonic ambient sound source (spass) dataset. *Data in Brief*, *50*, 109552. https://doi.org/10.1016/j.dib.2023.109552

Wang, B., Zhao, D., Lioma, C., Li, Q., Zhang, P., & Simonsen, J. G. (2020). Encoding word order in complex embeddings, In *8th international conference on learning representations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*, OpenReview.net. https://openreview.net/forum?id=Hke-WTVtwr

Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., & Gool, L. V. (2016). Temporal segment networks: Towards good practices for deep action recognition, In *Computer vision - eccv 2016 - 14th european conference, amsterdam, the netherlands, october 11-14, 2016, proceedings, part viii*, Springer. https://doi.org/10.1007/978-3-319-46484-8_2

Wang, L., Zhang, X., Su, H., & Zhu, J. (2024). A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis Machine Intelligence*, *46*(8), 5362–5383. https://doi.org/10.1109/TPAMI.2024.3367329

Wang, Z., Wang, D., Zhang, Y., & Xia, G. (2020). Learning interpretable representation for controllable polyphonic music generation, In *Proceedings of the 21th international society for music information retrieval conference, ISMIR 2020, montreal, canada, october 11-16, 2020*. http://archives.ismir.net/ismir2020/paper/000094.pdf

Wehrmann, J., & Barros, R. C. (2017). Movie genre classification: A multi-label approach based on convolutions through time. *Applied Soft Computing*, *61*, 973–982. https://doi.org/10.1016/j.asoc.2017.08.029

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. http://papers.nips.cc/paper%5C_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html

Wei, J., Yang, X., & Dong, Y. (2021). User-generated video emotion recognition based on key frames. *Multimedia Tools and Applications*, *80*(9), 14343–14361. https://doi.org/10.1007/S11042-020-10203-1

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, *3*, 1–40. https://doi.org/10.1186/S40537-016-0043-6

Weston, J., Dinan, E., & Miller, A. H. (2018). Retrieve and refine: Improved sequence generation models for dialogue, In *Proceedings of the 2nd international workshop on search-oriented conversational ai, scai@emnlp 2018, brussels, belgium, october 31, 2018*. https://doi.org/10.18653/v1/w18-5713

Whitelaw, C., Garg, N., & Argamon, S. (2005). Using appraisal groups for sentiment analysis, In *Proceedings of the 2005 ACM CIKM international conference on information and knowledge management, bremen, germany, october 31 - november 5, 2005*, ACM. https://doi.org/10.1145/1099554.1099714

Wiem, M. B. H., & Lachiri, Z. (2017). Emotion classification in arousal valence model using mahnob-hci database. *International Journal of Advanced Computer Science and Applications*, *8*(3). https://doi.org/10.14569/IJACSA.2017.080344

Williams, D., Kirke, A., Eaton, J., Miranda, E., Daly, I., Hallowell, J., Roesch, E., Hwang, F., & Nasuto, S. J. (2015). Dynamic game soundtrack generation in response to a continuously varying emotional trajectory, In *Audio engineering society conference: 56th international conference: Audio for games*. https://aes2.org/publications/elibrary-page/?id=17593

Williams, D., Kirke, A., Miranda, E. R., Roesch, E., Daly, I., & Nasuto, S. (2015). Investigating affect in algorithmic composition systems. *Psychology of Music*, *43*(6), 831–854. https://doi.org/10.1177/0305735614543282

Wingstedt, J., Berg, J., Liljedahl, M., & Lindberg, S. (2005). Remupp: An interface for evaluation of relations between musical parameters and perceived properties, In *Proceedings of the international conference on advances in computer entertainment technology, ace 2005, valencia, spain, june 15-15, 2005*, ACM. https://doi.org/10.1145/1178477.1178543

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., & Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, *abs/1910.03771*arXiv 1910.03771. http://arxiv.org/abs/1910.03771

Wu, S., Xu, J., Tai, Y.-W., & Tang, C.-K. (2018). Deep high dynamic range imaging with large foreground motions, In *Computer vision - ECCV 2018 - 15th european conference, munich, germany, september 8-14, 2018, proceedings, part II*, Springer. https://doi.org/10.1007/978-3-030-01216-8_8

Wu, S.-L., & Yang, Y.-H. (2020). The jazz transformer on the front line: Exploring the shortcomings of ai-composed music through quantitative measures, In *Proceedings of the 21th international society for music information retrieval conference, ISMIR 2020*. http://archives.ismir.net/ismir2020/paper/000339.pdf

Xu, B., Fu, Y., Jiang, Y.-G., Li, B., & Sigal, L. (2018). Heterogeneous knowledge transfer in video emotion recognition, attribution and summarization. *IEEE Transactions on Affective Computing*, 255–270. https://doi.org/10.1109/TAFFC.2016.2622690

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention, In *Proceedings of the 32nd international conference on machine learning, ICML 2015, lille, france, 6-11 july 2015*, JMLR.org. http://proceedings.mlr.press/v37/xuc15.html

Xu, Q., Huang, G., Yuan, Y., Guo, C., Sun, Y., Wu, F., & Weinberger, K. Q. (2018). An empirical study on evaluation metrics of generative adversarial networks. *CoRR*, *abs/1806.07755*arXiv 1806.07755. http://arxiv.org/abs/1806.07755

Yadav, A., & Vishwakarma, D. K. (2020). A unified framework of deep networks for genre classification using movie trailer. *Applied Soft Computing*, *96*, 106624. https://doi.org/10.1016/j.asoc.2020.106624

Yang, L.-C., Chou, S.-Y., & Yang, Y.-H. (2017). Midinet: A convolutional generative adversarial network for symbolic-domain music generation, In *Proceedings of the 18th international society for music information retrieval conference, ISMIR 2017, suzhou, china, october 23-27, 2017*. https://ismir2017.smcnus.org/wp-content/uploads/2017/10/226%5C_Paper.pdf

Yang, R., Wang, D., Wang, Z., Chen, T., Jiang, J., & Xia, G. (2019). Deep music analogy via latent representation disentanglement, In *Proceedings of the 20th international society for music information retrieval conference, ISMIR 2019, delft, the netherlands, november 4-8, 2019*. http://archives.ismir.net/ismir2019/paper/000072.pdf

Yi, Y., Zhou, J., Wang, H., Tang, P., & Wang, M. (2024). Emotion recognition in user-generated videos with long-range correlation-aware network. *IET Image Processing*, *18*(12), 3288–3301. https://doi.org/10.1049/IPR2.13174

Yoshida, Y., & Abe, M. (1994). An algorithm to reconstruct wideband speech from narrowband speech based on codebook mapping, In *The 3rd international conference on spoken language processing, ICSLP 1994, yokohama, japan, september 18-22, 1994*, ISCA. https://doi.org/10.21437/ICSLP.1994-412

Yu, Z., Yu, J., Fan, J., & Tao, D. (2017). Multi-modal factorized bilinear pooling with co-attention learning for visual question answering, In *IEEE international conference on computer vision, ICCV 2017, venice, italy, october 22-29, 2017*, IEEE Computer Society. https://doi.org/10.1109/ICCV.2017.202

Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks, In *Proceedings of the british machine vision conference 2016, BMVC 2016, york, uk, september 19-22, 2016*, BMVA Press. https://bmva-archive.org.uk/bmvc/2016/papers/paper087/index.html

Zaib, M., Zhang, W. E., Sheng, Q. Z., Mahmood, A., & Zhang, Y. (2022). Conversational question answering: A survey. *Knowledge and Information Systems*, *64*(12), 3151–3195. https://doi.org/10.1007/s10115-022-01744-y

Zander, M. F. (2006). Musical influences in advertising: How music modifies first impressions of product endorsers and brands. *Psychology of Music*, *34*(4), 465–480. https://doi.org/10.1177/0305735606067158

Zhang, H., & Xu, M. (2023). Recognition of emotions in user-generated videos through frame-level adaptation and emotion intensity learning. *IEEE Transactions on Multimedia*, *25*, 881–891. https://doi.org/10.1109/TMM.2021.3134167

Zhang, S., Dong, L., Li, X., Zhang, S., Sun, X., Wang, S., Li, J., Hu, R., Zhang, T., Wu, F., & Wang, G. (2023). Instruction tuning for large language models: A survey. *CoRR*, *abs/2308.10792*arXiv 2308.10792. https://doi.org/10.48550/ARXIV.2308.10792

Zhang, Y., Tian, Y., Kong, Y., Zhong, B., & Fu, Y. (2018). Residual dense network for image super-resolution, In *2018 IEEE conference on computer vision and pattern recognition, CVPR 2018, salt lake city, ut, usa, june 18-22, 2018*, Computer Vision Foundation / IEEE Computer Society. https://doi.org/10.1109/CVPR.2018.00262

Zhang, Z., Wang, L., & Yang, J. (2023). Weakly supervised video emotion detection and prediction via cross-modal temporal erasing network, In *IEEE/CVF conference on computer vision and pattern recognition, CVPR 2023, vancouver, bc, canada, june 17-24, 2023*, IEEE. https://doi.org/10.1109/CVPR52729.2023.01811

Zhang, Z., Zhao, P., Park, E., & Yang, J. (2024). MART: masked affective representation learning via masked temporal distribution distillation, In *IEEE/CVF conference on computer vision and pattern recognition, CVPR 2024, seattle, wa, usa, june 16-22, 2024*, IEEE. https://doi.org/10.1109/CVPR52733.2024.01219

Zhao, H., Gan, C., Rouditchenko, A., Vondrick, C., McDermott, J. H., & Torralba, A. (2018). The sound of pixels, In *Computer vision - ECCV 2018 - 15th european conference, munich, germany, september 8-14, 2018, proceedings, part I*, Springer. https://doi.org/10.1007/978-3-030-01246-5_35

Zhao, K., Li, S., Cai, J., Wang, H., & Wang, J. (2019). An emotional symbolic music generation system based on lstm networks, In *2019 ieee 3rd information technology, networking, electronic and automation control conference (itnec)*. https://doi.org/10.1109/ITNEC.2019.8729266

Zhao, S., Ma, Y., Gu, Y., Yang, J., Xing, T., Xu, P., Hu, R., Chai, H., & Keutzer, K. (2020). An end-to-end visual-audio attention network for emotion recognition in user-generated videos, In *Aaai 2020*, AAAI Press. https://doi.org/10.1609/AAAI.V34I01.5364

Zhou, B., Andonian, A., Oliva, A., & Torralba, A. (2018). Temporal relational reasoning in videos, In *Computer vision - eccv 2018 - 15th european conference, munich, germany, september 8-14, 2018, proceedings, part i*, Springer. https://doi.org/10.1007/978-3-030-01246-5_49

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., & Torralba, A. (2018). Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(6), 1452–1464. https://doi.org/10.1109/TPAMI.2017.2723009

Zhou, H., Hermans, T., Karandikar, A. V., & Rehg, J. M. (2010). Movie genre classification via scene categorization, In *Proceedings of the 18th international conference on multimedia 2010, firenze, italy, october 25-29, 2010*, ACM. https://doi.org/10.1145/1873951.1874068

Zhu, J., Sakurai, K., Togo, R., Ogawa, T., & Haseyama, M. (2024). Mmt-bert: Chord-aware symbolic music generation based on multitrack music transformer and musicbert. arXiv. https://doi.org/10.48550/arXiv.2409.00919

Zhu, X., Li, L., Liu, J., Peng, H., & Niu, X. (2018). Captioning transformer with stacked attention modules. *Applied Sciences*, *8*(5). https://doi.org/10.3390/app8050739

Zhuo, L., Wang, Z., Wang, B., Liao, Y., Bao, C., Peng, S., Han, S., Zhang, A., Fang, F., & Liu, S. (2023). Video background music generation: Dataset, method and evaluation, In *IEEE/CVF international conference on computer vision, iccv 2023*, IEEE. https://doi.org/10.1109/ICCV51070.2023.01433