Cross-Modal Musical Audio-Conditioned Image Synthesis

Lubomir Zelinsky *Supervised by:* Dr Bailin Deng
School of Computer Science and Informatics, Cardiff University
https://github.com/Lubober/CoverGen

Abstract

Recent advances in generative modeling have unlocked a vast amount of opportunity for domain-specific applications. This paper addresses the under-explored problem of musical audio-to-image synthesis, by developing CoverGen: a proof-of-concept pipeline, derived by cross-training StableDiffusion (Rombach et al. 2021), a text-to-image diffusion model with OpenL3 (Cramer et al. 2019), an audio embedding encoder model. The training was done on 19k+ audio-image pairs and performed in a parameter-efficient manner via Low Rank Adaptation (LoRA) (Hu et al. 2021), where a separate neural network is initialised and trained as a mapping bridge between the models. Furthermore, the paper describes the exploratory process of working towards the solution, and justifies the effectiveness of the methodology through the comparison of alternatives, as well as explains the intuition behind the solution so that it can be recreated and used for further development of the field.

1. Introduction

The difficulty of generating images from musical audios primarily comes from the ambiguity and subjectivity of their correlation. There is no singular image which can be a representation of a song, and it can be argued that a correlation can be mapped primarily through artistic sentiments. Therefore, CoverGen is an instance of generative AI art, which specialises in musical audio, and pays attention to sentiments which have been undermined in previous work.

2. Related Work

In order to understand how the solution discussed in this paper sits within the problem space of cross-domain AI art, it is crucial to firstly explain the problem itself, as well as the main advancements in it. This section aims to give a chronological overview of what steps have been taken to advance the field of AI art, and what has made it possible for this field to be skewed towards musical cross-modal image generation.

2.1 Early AI Image Generation

Early generative models were primarily general-purpose. Starting from Variational Autoencoders (VAE) (Kingma and Welling 2013), consisting of an encoder and a decoder. Trained on simple datasets such as the MNIST (Deng 2012) and small face datasets, the model learns to encode an image into a latent space by predicting a mean and variance, which is then fed into the decoder to reconstruct the image.

Generative Adversarial Networks (GANs) (Goodfellow et al. 2014) advanced the field, their architecture consisting primarily of a generator and a discriminator. The generator's purpose is to generate an image, and the discriminator's purpose is to try and identify whether the image that it's fed is real, or generated by the model. As the training goes on, the discriminator and the generator would progressively improve in tandem with their minimax objectives.

Diffusion models (Sohl-Dickstein et al. 2015) would then introduce the next step in generative modeling. Instead of a single pass encode-decode process which VAEs introduced, they would instead define an iterative forward noise-denoise process, in which the image would progressively

be corrupted by adding small amounts of Gaussian noise at each timestep. The model would then learn to invert that chain, also in an iterative step by step process. Such an approach would produce samples of significantly higher fidelity than VAEs.

At the time of their creation, the mentioned above generative models would strictly serve the role of proof-of-concept, however it is not long afterwards that these models would be utilised for further AI-art modeling advancements.

2.2 Text-To-Image

Text-to-image generation would then shortly be introduced by combining the previously mentioned generative models with text encoders.

Text encoders are a crucial step in cross modal communication. Since machine learning models are not directly able to understand text, encoders are used to convert textual data into machine readable numeric data, based on the features that the text possesses. The outputs would either be a single embedding (if sentence level), or a set of embeddings (for word-level attention).

Word2vec (Mikolov et al. 2013) introduced a simple, scalable neural method. It introduced a technique for word-level embeddings generation, with no regards for sentence-level context. (Sutskever et al. 2014) then would process all those embeddings token by token to encode that sequence into a single summary vector via a Recurrent Neural Network (RNN) (Elman 1990). Transformer (Vaswani et al. 2017) introduced self-attention which identifies which words are more important and pays more attention to them, which would evolve text encoders into a context-aware state. CLIP (Radford et al. 2021) would then teach text and image Transformers to share a common embedding space. Due to the richness of the dataset that it was trained on, it possesses zero-shot capability, and serves as a modern bridge between text and computer vision.

One of the first text-to-image generative models is believed to be alignDRAW (Mansimov et al. 2015). Built on top of DRAW (Gregor et al. 2015) (a VAE variant, which uses recurrent attention and canvas refinement to mimic the step-by step process of drawing), alignDRAW utilises an RNN text encoder to parse the feature vector into a VAE generator. The training was done on 80+k image-text pairs, with images being resized to 32×32px and text being limited to the first 5 captions. The generated images would possess abstract similarities with the caption, but would be blurry and of very low quality.

Scott Reed et al. (Reed et al. 2016) was the first to condition a Deep Convolutional GAN on text features encoded by an RNN. The model was generating sharper and higher-resolution samples, yet they were still of low resolution. Additionally, the model demonstrated an ability to disentangle style from content, allowing for applications like style transfer. Having shown a higher fidelity in generated outputs, further work would continue to incorporate GANs.

For the next couple years, the only advancements would be increments of existing GAN architecture. That would be until DALL-E (Ramesh et al. 2021) was introduced, which would push the field in a simple, yet impactful way. The primary distinction of the model was its scale, with a 12-billion parameter autoregressive transformer trained on a massive 250 million image-text pairs. The image generation in this work is a two-stage process, primarily driven by an autoregressive transformer, reconstructed by a discrete VAE. A pre-trained contrastive model would be used for ranking generated images, and would select the best candidate.

After DALL-E, the text-to-image paradigm shift was towards the direction of diffusion models. Starting with GLIDE (Nichol et al. 2021), and then following by DALL-E 2 (Ramesh et al. 2022), Google Imagen (Saharia et al. 2022), StableDiffusion (Rombach et al. 2021), which is

used in this particular work and is explained in more detail later on.

Natural Language Processing (NLP) is the field of computer science and linguistics which focuses on improving the understanding of languages. NLP has been studied since the early 1950s (Turing 1950) and has seen massive development ever since, especially with the transformer-based Large Language Models (Devlin et al. 2018; Brown et al. 2020; Bai et al. 2022). The richer understanding of textual features is able to lead to much more successive improvements in text-to-image synthesis, due to much richer and more precise text embeddings, and thus, to much better image generation. The progress in text-to-image synthesis is directly linked to the progress of NLP.

2.3 Audio-To-Image

Following a similar methodology, audio-to-image models have been implemented. Audio files, however, possess fundamentally different properties compared to text, and extracting useful information from audio files primarily falls under signal processing.

The first Auto Audio Captioning (AAC) began with RNN-based encoder-decoder (Drossos et al. 2017), operating primarily on log-mel spectrograms to generate textual captions. The model was trained on >150k recordings, each associated with a single textual description. The results of the evaluation have shown that the proposed method's performance is "well above of a random guess of a sequence of words".

Early work primarily used GANs, inspired by the first audio-to-image pipeline (Chen et al. 2017) which involves processing an audio file into a log-mel spectrogram and encoding it into an embedding space using a Convolutional Neural Network (CNN), which treats the spectrogram as an image. The dataset that the model was trained on involved pairs of images and sounds of musical performances of different instruments, composed by the researchers themselves. The model's use case was in identifying which instrument an audio contained, and what exact pose the performer of the instrument was in. The generated images were of rather poor quality, however the evaluation showed rather accurate results.

Speech2Face (Oh et al. 2019) would attempt to generate a corresponding face image from audios of human speech. Utilising a pre-trained face decoder network, Speech2Face would train its own spectrogram-powered voice encoder, and would use YouTube videos to construct a dataset. The model would generally generate coherent and relevant results, with occasional mistakes due to outliers in the dataset.

AudioToken (Yariv et al. 2023) used text-to image StableDiffusion and a pre-trained audio embedder. Unlike any other work, it doesn't fine-tune the generative model, and instead it updates only the weights of the linear and attentive pooling layers within the Embedder network, which is their pre-trained audio captioning model. The Embedder layer then produces a dedicated audio token which is concatenated onto a textual latent space. The combined, concatenated tensor of representations is then fed into the generative model. Trained on a video dataset, the model performed considerably well. This differs from Wav2CLIP (Wu et al. 2022), which utilises CLIP's vision and trains an audio encoder from scratch. Both the models have developed the audio to image solution space, yet focused primarily on environmental sounds.

2.4 Music-To-Image

Generating images from music involves primarily extracting the emotional sentiment from songs.

Early work (Chen et al. 2008) focuses on performing acoustical feature analysis to estimate the emotional probability distribution of a novel music emotion model based on the updated Hevner's 8 emotion groups (Wu and Jeng 2008).

The research does not generate images, but instead performs feature-based classification by selecting music features using F-scores and building a multi-class classifier using Support Vector Machines (SVM) to assign one of the eight predefined emotional labels to the song, and linking them to existing images. The research serves as a feature extractor and did not generate images, but it would then inspire the next advancement in music-to-image modeling.

One of the first music-oriented audio-to-image pipelines (Qiu and Kataoka 2018) would use Short Time Fourier Transform to extract acoustic sentiment from musical data, which would then be aggregated into a 1024 dimension feature vector via the CNN model. Using an existing image feature extraction model AlexNet, the images from the dataset would be represented by a 1024 dimensional embedding space. The training would then aim to learn the correlation between these embeddings by computing their inner product, and would update the audio encoder to match the image embeddings. Once the audio encoder was trained to match the image encoder, a Conditional Deep Convolutional GAN (Radford et al. 2015) would be trained by using the audio embeddings as conditions. The dataset that the model was trained on, attempted to capture the correlation between music and 4 types of environmental scenes. The results were of low quality, but were accepted by the conducted survey study.

Deepsing (Passalis and Doropoulos 2021) used techniques similar to those presented in the earlier work (Chen et al. 2008) for emotional sentiment musical feature extraction. The pipeline utilises an audio attribute estimator, trained to regress the valence and arousal of audio stimuli, which are key sentiment attributes. Features such as Mel Frequency Cepstral Coefficients (MFCC), Chroma Energy Normalised (CENS) features, and tempogram features are extracted from each 500ms frame and fed into a neural regressor, which puts it into an attribute space. A pre-trained BigGAN model (Brock et al. 2018) is used for 512×512px image generation, and a novel translator is trained to address the challenge of an "unstable mapping" between the sentiment space and the complex GAN generator space. Trained on the DEAM dataset (Soleymani et al. 2018), the quantitative results have shown that the model has outperformed its control method to prove its success.

Despite the development in cross-domain image generation, the problem of generating images from musical audio files remains underexplored, particularly due to the scarcity of musical datasets. A shortage of peer reviewed articles further suggests the limitation in the research and indicates a gap in the research, which CoverGen attempts to fill.

2.5 Methodology Components

The solution described within this paper, CoverGen, exists specifically within the musical audio-to-image problem space and attempts an alternative approach, not found in previous work, to achieve the solution. Leveraging a more modern set of tools and techniques, the main components of the methodology and their comparative efficiency are described in this section.

2.5.1 OpenL3

OpenL3 (Cramer et al. 2019) is an open-source deep audio embedding model based on the Look, Listen and Learn (L3-Net) architecture. The training was done via a self-supervised audio-video correspondence learning, meaning it learned to produce similar embeddings for audio and video that occur together in videos. In this solution, OpenL3 was used primarily for generating embeddings from audio files, which would then be fed into a generative model.

OpenL3 was chosen as the audio feature extractor due to its strong performance in capturing musical information and its open availability. Studies have shown that OpenL3's embeddings outperform earlier audio embeddings like Google's VGGish or SoundNet on sound recognition tasks.

The OpenL3 model is domain-specific. CoverGen uses the model trained on musical content, which is ideal for musical audio analysis (as opposed to a model trained on environmental sounds). This specialisation helps the embedding capture musical attributes (rhythm, timbre, melody) more effectively.
The initial approach that was considered for the solution was to use an audio to text model to generate intermediary textual content. The two models that were tested were MusCaps (Manco et al. 2021) and MusicBERT (Zeng et al. 2021). The idea was then to train an adapter between the extracted text from the audio and extracted text from an image by using BLIP-2 (Li et al. 2023). The described approach would be much less effective because textual embeddings lose a lot of musical sentiments, while OpenL3 treats audio as a log-mel spectrogram and is capable of capturing much more sonic features. Such features are very essential, particularly in capturing the character of a song. The final solution does not have any intermediary conversions and does not lose any sentiment, allowing the models to communicate directly. This can allow the solution to learn its own patterns and develop an understanding of artistic sentiments through pattern recognition.

2.5.2 Stable Diffusion

At the heart of the image generation is Stable Diffusion (Rombach et al. 2022), a state-of-the-art text-to-image latent diffusion model. The model is built on several layers, and is open-source, which enables re-training and re-purposing parts of the model as well as fine-tuning it for a particular purpose.

The model uses a BERT (Devlin et al. 2018) text encoder, and a VAE for image encoding/decoding. In between them is where the actual diffusion process happens. A clear distinction is in place, that diffusion does not operate on images, but instead operates on latent spaces encoded by the VAE. This saves a lot of computation.

Operating strictly on latent spaces, the diffusion model consists of a Fixed Markov Chain forward component and a U-Net (Ronneberger et al. 2015) reverse diffusion component. A Markov Chain is responsible for adding Gaussian noise progressively onto a latent space, and the U-Net then learns to apply de-noising iteratively. As a result, the U-Net is then able to apply denoising from scratch, and that is what is used in the generative process.

So when StableDiffusion runs, it first encodes the text that it is fed into a feature vector. It then uses the feature vector as a conditioning input for the U-net, which refines a noisy latent space into a cleaner latent space. The latent representation is then fed into the VAE to be decoded, and that is where the final image is generated.

Diffusion models have rapidly become the preferred approach for generative image modeling due to their stability in training and high output fidelity. Research (Dhariwal and Nichol 2021) demonstrates that diffusion models can achieve image sample quality surpassing GANs, attaining better data distribution coverage and superior FID scores.

The first attempts to fine-tune StableDiffusion were unsuccessful, due to a resource shortage. The model was initially trained on around 1.5b text-image pairs, with hundreds of hours of high-end GPU computation, and retraining it on 19k data points was not proving to be successful. A more parameter-efficient methodology had to be implemented.

2.5.3 Low Rank Adaptation (LoRA)

Low Rank Adaptation (Hu et al. 2021) is a parameter-efficient large models fine-tuning methodology which is able to massively reduce computational overhead associated with training as a result of the reduction in trainable parameters. The way that LoRA is implemented is the

following: the model's trainable parameters are frozen, meaning that they remain unaffected during training; a set of trainable matrices is then injected on top of the model; the newly injected parameters are trained. Once the training is completed, the injected trainable parameters are kept, and they skew the direction of the entire model's generation towards their learning.

The way that LoRA is utilised in this work, however, differs from its normal use case, as instead of inserting the trainable parameters into the model, a separate standalone adapter model is trained, to bridge the OpenL3 embeddings with the StableDiffusion generative model. The adapter in theory could be portable, and could theoretically be used for using other generative models, although that hasn't been explicitly proven yet. The adapter is explained in more detail later on, in the methodology section.

3. Method

The training was done on 19k+ audios and their corresponding image pairs. Taken from a Spotify metadata dataset, an Extract Transform Load (ETL) pipeline had to be constructed to cleanse the dataset of unnecessary/faulty data. The audio files would be processed with the use of OpenL3, to generate feature vectors which would be used in the training pipeline. The training would then involve developing a lightweight cross-modal independent mapping adapter to act as a bridge between Stable Diffusion's U-Net and OpenL3 latent spaces. The training was done in a parameter-efficient way with the help of low-rank adaptation (LoRA) techniques, and yielded promising results.

For the inference pipeline to be functional, a solution was developed which is fed an audio file, and finds its most eventful 30s segment by using Root Mean Square (RMS) energy and spectral contrast. An OpenL3 embedding would then be generated from the segment, and would be evaluated via StableDiffusion, with the use of the newly trained mapper. The generated image by the model, however, would usually be noisy, so an img2img StableDiffusion model would be used for de-noising.

3.1 Data Handling

Using the Kaggle platform, a dataset containing historical metadata from a music streaming service, Spotify (Singh 2023), was used. The dataset was in the csv format, containing the following columns:

```
Name, artist, spotify_id, preview, img, danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, acousticness_artist, danceability_artist, energy_artist, instrumentalness_artist, liveness_artist, speechiness_artist, valence_artist
```

Simplicity is the main goal of the solution, thus the only two columns that were used were "preview", and "img". For training ML models, the dataset that is used is one of the most key factors affecting their behaviour, thus the intuition behind using a streaming service dataset is that there is a degree of ground truth behind the correlation of the musical files and their corresponding album covers. The theory is that while the correlation may not be fully logical, it can be perceived as artistic. This can lead to ML models to be able to recognise new artistic sentiments, and lead to a whole new artistically-led generative behaviour.

The dataset needed to be cleansed, as it was far from a ready-to-use state. The cleansing involved removing unnecessary columns, keeping only the "preview" and the "img" columns. A lot of rows would contain null entries, thus it was necessary to get rid of those. The data points did not contain actual images, nor audios, instead they contained their URLs. Using "wget", the .mp3 and the .png files were obtained and put into their respective folders, named after their respective id's.

Next, once the audios were aggregated, OpenL3 was used to generate feature vectors from them. The choice was to pre-process them, primarily for efficiency purposes, as each embedding generation of 30s would take about 10-20 seconds of computation. The pre-processing would take about

a week of computation, as the size of the cleansed dataset was of about 19k+.

Having finished the processing, a csv file was generated, containing the locations of the data points, with a structure:

```
id,image_path,embedding_path
```

A 80/10/10 split would then be applied to the dataset randomly to obtain the train (15215 entries),validation (1902 entries) and test (1902 entries) datasets. There was now sufficient data to begin the training.

3.2 Model Architecture

The training was done on the aggregated data, where a torch dataset is initially constructed by utilising the *PairedDataset()* class using the csv file containing the directories of the .png and the .npy files. The appropriate image resizing is applied to images, and using the class, the 3 instances of datasets (train, test, valid), torch datasets are initialised. Using the *pad_audio_embeddings()*, the OpenL3 embeddings are padded to match the maximum batch sequence length.

3.2.1 Attention Pooling

Attention pooling is an essential process which is applied to each encoded audio file. Each audio sample is initially represented as a sequence of 512-dimensional embeddings over time (e.g., shape [T, 512], where T is the number of time steps). These embeddings capture different features of the audio signal at different moments in time — such as timbre, rhythm, and harmonic content. However, the downstream model (Stable Diffusion) expects a fixed-size vector to condition the image generation process. For each sample i and time step t, the attention score $s_{i,t}$ is computed as:

$$s_{i,t} = w^\top \tanh(W E_{i,t} + b)$$

The attention weights $\alpha_{i,t}$ are then obtained via a softmax function:

$$\alpha_{i,t} = \frac{\exp(s_{i,t})}{\sum_{t'=1}^{T_i} \exp(s_{i,t'})}$$

The aggregated embedding $\tilde{E}_i$ is calculated as a weighted sum:

$$\tilde{E}_i = \sum_{t=1}^{T_i} \alpha_{i,t} \cdot E_{i,t}$$

As a result, the single pooled vector is $\tilde{E}_i \in \mathbb{R}^{512}$, which allows the model to focus only on the most informative parts of the audio.

In the later logic of the implementation, this comes with certain tradeoffs, as StableDiffusion generally takes 77 tokens in, and the attention pooling reduces the number of tokens to 1. How the other 76 tokens are filled is by duplicating the 1 token 76 times, which enables the training to be much more computationally cheap. Although, the tradeoff could potentially be loss of fine-grained info, as there is no temporal structure in the embeddings.

3.2.2 LoRA Mapper

Through a similar logic of LoRA methodology, the U-Net of Stable Diffusion is kept frozen. However, rather than injecting low-rank trainable matrices directly into the U-Net (as is typical in LoRA-based fine-tuning), this implementation introduces a separate trainable module, *LoRAAudioToImageMapper()*.

This adapter is a separate neural network model which acts as a bridge between the pooled OpenL3 audio embeddings and the conditioning mechanism of the U-Net in Stable Diffusion. Specifically, it transforms 512-dimensional audio features into 768-dimensional embeddings that are compatible with the U-Net's cross-attention layers, allowing the audio signal to guide image generation.

3.2.2.1 LoRALinear Module

The LoRALinear module applies the LoRA logic to the U-Net of StableDiffusion.

In conventional feedforward neural networks, a fully connected layer performs the following operation:

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

where:

- $\mathbf{x} \in \mathbb{R}^d$ *is the input vector,*
- $\mathbf{W} \in \mathbb{R}^{d' \times d}$ *is the weight matrix,*
- $\mathbf{b} \in \mathbb{R}^{d'}$ *is the bias vector, and*
- $\mathbf{y} \in \mathbb{R}^{d'}$ *is the output.*

In LoRA, the original weight matrix $\mathbf{W}$ *is frozen and is not updated during training. Instead, a trainable low-rank update* $\Delta\mathbf{W}$ *is introduced, such that the effective weight becomes:*

$$\mathbf{W}_{\text{eff}} = \mathbf{W} + \alpha \cdot \Delta\mathbf{W}$$

The update $\Delta\mathbf{W}$ *is factorised into the product of two smaller matrices:*

$$\Delta\mathbf{W} = \mathbf{B}\mathbf{A}, \quad \text{where} \quad \mathbf{A} \in \mathbb{R}^{r \times d}, \quad \mathbf{B} \in \mathbb{R}^{d' \times r}$$

Here, $r \ll \min(d, d')$ *denotes the rank of the adaptation, and* $\alpha \in \mathbb{R}$ *is a scalar hyperparameter used to scale the contribution of the adaptation. Substituting into the forward pass, the overall transformation becomes:*

$$\mathbf{y} = (\mathbf{W} + \alpha \cdot \mathbf{B}\mathbf{A})\mathbf{x} + \mathbf{b}$$

This formulation ensures that the base model remains unchanged, and only the low-rank matrices $\mathbf{A}$ *and* $\mathbf{B}$ *are updated during training. As a result, the total number of trainable parameters is reduced from* $d' \cdot d$ *(in a full-rank linear layer) to* $r \cdot (d + d')$*, which is a significant reduction when* $r \ll d, d'$*.*

This parameterisation enables to minimise the computational overhead, making it especially suitable for fine-tuning under limited resource constraints. In this particular use case it also works because the dataset that is used for training is much smaller from the original 1.5bn entries dataset that StableDiffusion was trained on, thus initial model integrity is to be preserved.

3.2.2.2 Residual Blocks

The *ResidualBlock()* module in this work is a fully connected residual learning unit designed to refine the internal feature representation while preserving input information. It consists of two linear layers with ReLU activation and dropout in between, followed by layer normalisation and a residual connection. Formally, for an input vector $\mathbf{x} \in \mathbb{R}^d$, the block computes:

$$\mathbf{y} = \text{LayerNorm}(\mathbf{x} + \text{Linear}_2(\text{Dropout}(\text{ReLU}(\text{Linear}_1(\mathbf{x})))))$$

where each component is defined as follows:

1. ***Linear Transformation***

The linear layers are affine projections:

$$\text{Linear}_i(\mathbf{x}) = \mathbf{W}_i\mathbf{x} + \mathbf{b}_i$$

Where

- $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d \times d}$ *are learned weight matrices.*
- $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^d$ *are learned bias vectors.*

2. ***ReLU Activation***

The ReLU function applies a non-linearity element-wise:

$$\text{ReLU}(x_j) = \max(0, x_j) \quad \text{for each component } j \in \{1, \ldots, d\}$$

This introduces non-linear capacity to the block and ensures gradient flow for positive activations.

3. ***Dropout Regularisation***

Dropout applies stochastic masking during training:

$$\text{Dropout}(\mathbf{h}) = \mathbf{m} \odot \mathbf{h}, \quad \mathbf{m} \sim \text{Bernoulli}(p)$$

Where $\mathbf{h} \in \mathbb{R}^d$ *is the input vector and* $\odot$ *denotes element-wise multiplication. The mask* $\mathbf{m}$ *is sampled such that each component is independently zeroed with probability* $1 - p$*.*

4. ***Layer Normalisation***

Layer normalization normalises the summed residual $\mathbf{x} + \mathbf{z}$ *across its feature dimension:*

$$\text{LayerNorm}(\mathbf{v}) = \frac{\mathbf{v} - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \gamma + \beta$$

Where:

- $\mu = \frac{1}{d}\sum_{j=1}^{d} v_j$ *and* $\sigma^2 = \frac{1}{d}\sum_{j=1}^{d}(v_j - \mu)^2$
- $\gamma, \beta \in \mathbb{R}^d$ *are learnable scaling and bias parameters*
- ϵ *is a small constant for numerical stability.*

This architecture allows the model to learn residual functions $\mathcal{F}(\mathbf{x}) = \mathbf{y} - \mathbf{x}$, enabling better gradient flow and facilitating a more stable and effective optimisation.

3.2.2.3 LoRAAudioToImageMapper

The LoRAAudioToImageMapper module serves as the core trainable network responsible for mapping pooled audio embeddings (of dimension 512) into a 768-dimensional conditioning vector suitable for input into Stable Diffusion's U-Net. It is a shallow feedforward neural network composed of LoRA-enhanced linear projections and residual refinement layers.

The architecture has three main stages:

1. ***Initial Projection***

The input vector $\mathbf{x} \in \mathbb{R}^{512}$ *is first transformed via a low-rank adaptation (LoRA) linear layer:*

$$\mathbf{h}_0 = \text{LayerNorm}(\text{ReLU}(\text{LoRALinear}_{\text{init}}(\mathbf{x})))$$

This projects the 512-dimensional audio embedding into a higher-dimensional latent space (typically 1024) using a frozen base weight and a trainable low-rank update. Layer normalisation and ReLU activation are applied to stabilise the feature space.

2. ***Residual Refinement***

The transformed vector $\mathbf{h}_0 \in \mathbb{R}^{1024}$ *is then passed through a sequence of* N *residual blocks* $\mathcal{R}_1, \ldots, \mathcal{R}_N$*, each designed to refine features while preserving the original signal:*

$$\mathbf{h}_N = \mathcal{R}_N(\ldots \mathcal{R}_1(\mathbf{h}_0)\ldots)$$

Each residual block follows the structure defined in Section 3.2.2.2.

3. ***Final Projection***

Finally, the output is projected to the target conditioning dimensionality (768) via another LoRA linear layer:

$$\mathbf{y} = \text{LoRALinear}_{\text{final}}(\mathbf{h}_N)$$

The resulting $\mathbf{y} \in \mathbb{R}^{768}$ *serves as the encoder hidden state fed into the U-Net of the diffusion model for image generation.*

This architecture balances expressivity and parameter efficiency. By incorporating **LoRA-based projections**, it enables efficient fine-tuning by updating only low-rank factor matrices, leaving the large backbone weights frozen. The inclusion of residual blocks ensures the model can perform iterative refinement while avoiding vanishing gradients.

3.3 Training Procedure

The StableDiffusionPipeline is loaded from the *runwayml/stable-diffusion-v1-5* checkpoint. Components such as the VAE, the U-Net and the Scheduler are extracted and frozen, as only the *LoRAAudioToImageMapper* and *AttentionPooling* are trainable. The selected optimiser is *adam*, with a learning rate of 10^{-4}. A *ReduceLROnPlateau* scheduler reduces learning rate by a factor of 0.5 if validation loss does not improve for 2 epochs. The number of epochs is set to a number between 10 and 30, where an improvement in the loss hasn't been seen for 5 epochs.

3.3.1 Training Loop

For each batch in the training set, the following sequence is applied:

1. ***Data Preparation***

The model receives a batch of RGB images $\mathbf{I} \in \mathbb{R}^{B\times 3\times 512\times 512}$ and corresponding audio embeddings $\mathbf{A} \in \mathbb{R}^{B\times T\times 512}$, where T is the (variable) temporal length of the OpenL3 embedding. Embeddings are padded across the batch for uniformity.

2. Attention Pooling

The audio embeddings are temporally pooled using an attention mechanism to obtain a fixed-size vector $\mathbf{a}_{\text{pooled}} \in \mathbb{R}^{B\times 512}$, which serves as a summary representation of the audio content.

3. Adapter Mapping

The pooled audio is passed through the *LoRAAudioToImageMapper*, producing a 768-dimensional conditioning vector for each sample:

$$\mathbf{c} = f_{\text{adapter}}(\mathbf{a}_{\text{pooled}}) \in \mathbb{R}^{B\times 768}$$

This vector is then repeated along the token dimension to form the conditioning tensor expected by Stable Diffusion:

$$\mathbf{C} = \text{Repeat}(\mathbf{c}, 77) \in \mathbb{R}^{B\times 77\times 768}$$

4. Latent Noise Prediction

Each image $\mathbf{I}$ is encoded into a latent representation $\mathbf{z} \in \mathbb{R}^{B\times 4\times 64\times 64}$ using the frozen VAE encoder, scaled by a fixed factor. Random Gaussian noise $\boldsymbol{\epsilon}$ is added at a randomly sampled timestep t. The U-Net is conditioned on both the noisy latent $\tilde{\mathbf{z}}$ and the conditioning vector $\mathbf{C}$, and attempts to predict the added noise:

$$\hat{\boldsymbol{\epsilon}} = f_{\text{U-Net}}(\tilde{\mathbf{z}}, t, \mathbf{C})$$

5. Loss Computation and Backpropagation

The objective is to minimise the mean squared error between the predicted noise and the true noise:

$$\mathcal{L} = \frac{1}{B}\sum_{i=1}^{B} \|\hat{\boldsymbol{\epsilon}}_i - \boldsymbol{\epsilon}_i\|_2^2$$

Gradients are computed and applied only to the adapter parameters (using gradient clipping), while all other components remain frozen.

6. Validation

After each epoch, the model is evaluated on the validation set using the same process, but without computing the gradients. The average validation loss is tracked to monitor generalisation. Sample reconstructed images are decoded from latent space using the VAE decoder and logged to TensorBoard for qualitative inspection.

3.3.2 Post-Training Evaluation

Using the disjointed test dataset, the model is evaluated using the same principal as in the training and the validation. It serves as an unbiased estimate of how well the adapter generalises to novel input data.

Following the evaluation, training and validation loss curves are plotted to offer a visual representation of the model's convergence behaviour and generalisation performance across epochs.

3.4 Inference Pipeline

Having done the training, an inference pipeline is constructed, which performs the following steps:

1. Chorus Extraction

To ensure that the generated image reflects the most musically salient portion of a track, a 30s segment is extracted. This feeds the model data in the format that it is used to handling and reduces computation. The extraction is designed to approximate the chorus, which is the part that carries the core thematic and emotional content of a song.

The approach adopts a sliding window strategy over the full audio waveform. For each window, a composite salience score is calculated using two audio features: Root Mean Square (RMS) energy and Spectral Contrast. These are computed using the *Librosa* signal processing library.

- **Root Mean Square (RMS) Energy**:

RMS energy measures the average power of the waveform over time. For a signal $y = [y_1, y_2, \ldots, y_N]$, the RMS is defined as:

$$\text{RMS}(y) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} y_i^2}$$

This value captures perceived **loudness**, allowing the system to prioritise dynamically intense passages.

- **Spectral Contrast**:

Spectral contrast quantifies the difference between the peaks and valleys of the log-magnitude spectrum across multiple frequency bands. It is defined as:

$$\mathrm{SC}(y) = \frac{1}{B}\sum_{b=1}^{B}(S_{\text{peak}}^{(b)} - S_{\text{valley}}^{(b)})$$

where B is the number of frequency bands. High spectral contrast indicates a rich harmonic structure, often found in complex and engaging musical sections.

For each window, the mean RMS and mean spectral contrast are multiplied to form a composite salience score:

$$\mathrm{Score}(y) = \mathrm{mean}(\mathrm{RMS}(y)) \times \mathrm{mean}(\mathrm{SC}(y))$$

The window with the highest score is selected as the **target chorus segment** and is extracted as an mp3.

2. OpenL3 Embedding Generation

The mp3 file is then fed into OpenL3, and a feature vector is derived from it, using the same configuration, as one in the training.

3. Image Generation

During inference, the Stable Diffusion model and the LoRA-based adapter are set to evaluation mode, disabling stochastic operations and gradient updates. Unlike training and validation, no noise is added and no loss is computed. Instead, the OpenL3 embedding $\mathbf{z}_{\text{audio}} \in \mathbb{R}^{512}$ is projected into a 768-dimensional latent vector via the trained adapter:

$$\mathbf{z}_{\text{mapped}} = f_{\text{adapter}}(\mathbf{z}_{\text{audio}})$$

This vector is repeated across 77 tokens to produce a conditioning tensor $\mathbf{Z}_{\text{cond}} \in \mathbb{R}^{77\times768}$ which replaces the usual text embeddings in the diffusion model. The pipeline then performs a single forward pass to generate an image semantically aligned with the input audio.

4. Image Refinement

To improve visual fidelity and reduce artifacts, the generated image is passed through an image-to-image diffusion process using the Stable Diffusion XL Refiner (v1.0). This model performs a denoising operation conditioned on both the input image and a textual prompt, acting as a post-processing step to enhance detail and consistency.

Formally, given an initial image $\mathbf{I}_{\text{init}}$, the process applies a controlled amount of noise (via latent interpolation) and then denoises it using the same diffusion principles as the base model. The transformation can be summarised as:

$$\mathbf{I}_{\text{refined}} = \mathrm{Refiner}(\mathbf{I}_{\text{init}}, \mathrm{prompt}, \mathrm{strength})$$

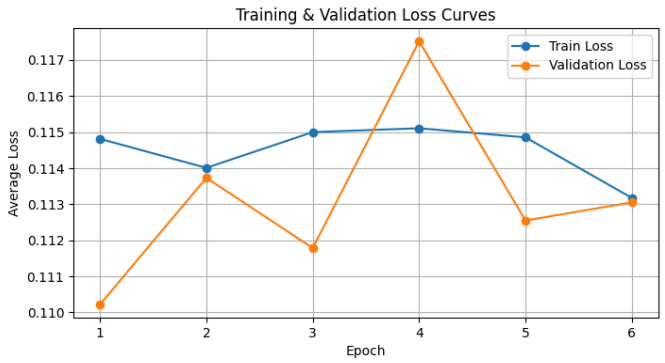
Where:

- prompt is a fixed textual prompt (e.g., *"album cover high resolution"*) used to guide the refinement
- $\mathrm{strength} \in [0, 1]$ controls how strongly the original image is altered.

A strength of 0.5 provides a balance between preserving the structure of the initial image and allowing the model to introduce photorealistic improvements. The output $\mathbf{I}_{\text{refined}}$ is the final high-quality album cover image.

4. Results and Evaluation

Having attempted to train the adapter on several attempts, the following curve was achieved on the best attempt:



Training was limited due to both practical constraints and hardware interruptions. Each epoch required at least 8 hours (usually much more), and the training was forcefully interrupted on multiple attempts by an automated Windows system update that would forcefully restart the machine. This unfortunate timing prevented the completion of additional epochs within available time constraints.

The curve presents a point which is far from convergence, with training loss decreasing from 0.115 to 0.113 and validation loss declining from 0.110 to 0.112. The consistent proximity between training and validation losses (gap < 0.003) indicates the model generalises well without signs of overfitting. Notable fluctuations in validation loss, particularly the spike at epoch 4, reflect the natural variability inherent in stochastic training when evaluated on smaller validation batches.

Despite the limited training, the generated results were of higher fidelity than anticipated, and show potential given a more complete training. Examples of generated images can be seen in the appendix.

4.1 Qualitative Assessment

Upon evaluating a participant user survey of 10 people, an agreement rate of ~52% was achieved. The survey contained 15 songs and their corresponding generated album covers, which the participants would rank from 1 to 5 based on how well the image fits the song. The results have shown that the model is able to generate artwork that is semantically coherent to the musical audio files the majority of times. The study has also revealed that the genres that scored the best are more mainstream genres like rock, hip-hop and pop, with less mainstream genres like classical, experimental and electronic, being generally perceived as less coherent. A result of 52% is technically a majority, but suggests that there is room for improvement. Due to limitations that are further discussed, the results are of a solution which is not fully realised, and the fact that the prototype is able to get a majority of positive votes, indicates that the pipeline is functional.

4.2 Reflection On Results

The model is able to generate its own visualised artistic interpretations of songs. The output of the model is unpredictable, and can generally lead to very interesting visuals.

The abbreviated 6-epoch training schedule represents the most significant constraint on performance. With the model stopping well before convergence, many observed limitations, including inconsistent genre representation and occasional generation of incomprehensible text, can be attributed to insufficient learning time. The fact that meaningful audio-visual mappings emerge within this truncated timeframe suggests potential.

Beyond training duration, the following dataset limitations may have affected the training:

High Duplicate Rate: Multiple songs in the same album sharing identical album artwork create a one-to-many ratio in the training data. This redundancy may cause the model to prioritise album-level aesthetics over song-specific nuances, explaining why some outputs default to generic cover-art styles rather than capturing unique track characteristics.

Genre & Style Bias: Popular artists and large albums dominate the dataset, skewing the model toward mainstream visual conventions. This imbalance particularly impacts performance on unconventional genres like electronic, cloud-rap, or experimental ambient, where the model defaults to established cover-art tropes rather than genre-appropriate aesthetics.

Demographic Bias: The model occasionally makes erroneous assumptions about artist ethnicity based on musical characteristics alone, thus for artists like Yung Lean and Bladee, it is prone to making mistakes. This highlights the need for carefully curated, bias-aware training data.

Text Generation Artifacts: The persistent issue of incomprehensible text in generated images suggests that text-heavy album covers create visual complexity that the current architecture cannot adequately reproduce. This represents a clear technical limitation requiring architectural improvements.

5. Conclusions and Future Work

This study investigated direct audio-to-image synthesis through parameter-efficient fine-tuning of latent diffusion models. The LoRA-based adapter achieved semantic alignment between OpenL3 audio embeddings and Stable Diffusion's conditioning space, enabling image generation from musical inputs without text intermediaries. Training MSE loss decreased from 0.115 to 0.113 across six epochs, with validation loss tracking closely at 0.110-0.112, indicating stable convergence, and showing potential for further training.

The qualitative user study has revealed that the model is able to make fitting artwork the majority of times. During the evaluation of the model, several weaknesses were found and are to be addressed for the pipeline to be considered more than a proof-of-concept.

5.1 Dataset Enhancements

The primary limitations of the current model can be traced to dataset characteristics. Future iterations would benefit from addressing these limitations by:

1. Eliminating duplicate album-level artwork
2. Balancing genre representation by aggregating a higher versatility of genres
3. Implementing fairness-aware techniques to mitigate demographic biases in the training data
4. Developing text-aware OCR-based pre-processing to identify textual contents, and account for them in the training
5. Account for the metadata of songs during the training to help provide more information

5.2 Enhanced Representation of Track-Level Semantics

The current OpenL3 embeddings could be augmented with complementary feature representations capturing different aspects of musical content. A way could potentially be found to incorporate track metadata into the training of the model, to allow it to pick up on more details, and allowing it to learn more. The lyrics of songs can also potentially be extracted, improving the coherence even more. Doing so would highly increase the complexity of the model, but would potentially enable a higher fidelity of results.

5.3 Extended Training With Cloud Computing

The most pressing improvement would be extended training duration using cloud computing resources. The current 6-epoch limitation, imposed by local hardware constraints, prevented the model from converging. The loss analysis shows that training was stopped well before the optimal point, with validation metrics suggesting significant headroom for improvement. An extended training with the use of cloud computing would address the primary quantitative limitation identified in the

evaluation, potentially elevating the system to production-ready standards for music-driven image synthesis.

6. Reflection on Learning

This project marked a significant progression in my machine learning capabilities. The initial data collection phase exposed practical challenges absent from academic literature. The data aggregation has turned out to be a much more demanding task than initially anticipated, as finding a dataset for such a task is not a common occurrence. Further, the extraction of the data and the pre-processing has taught me a lot about how messy data can be in a real-life scenario and how to cleanse it properly. These early experiences fundamentally shaped my understanding that machine learning success depends critically on data integrity and preprocessing rigour.

Evaluating alternative approaches broadened my technical perspective. Reading through literature has improved my technical understanding of the field of Machine Learning and has broadened my knowledge of what existing solutions have for similar tasks as mine. The existing solutions have skewed my solution towards a functional, parameter efficient one and I have gained a vast amount of experience from it.

The LoRA implementation phase challenged my programming abilities most significantly. Translating research papers into working code required developing new debugging skills, particularly around tensor dimensionality mismatches and numerical precision issues.

Upon the evaluation of the model, I have discovered numerous dataset biases, which has given me actionable insights for what to consider when working with data in the future. The subjectivity in cross-modal tasks challenged the evaluation of it, as it was virtually impossible to get a quantitative evaluation of the model's effectiveness.

By project completion, I had developed from someone who understood concepts superficially to someone capable of designing, implementing, and critically evaluating a machine learning pipeline.

This experience established technical skills and a methodological approach to machine learning problems. These lessons will fundamentally inform my future work in Machine Learning Engineering and any other data related role, as well as in my postgraduate studies.

References

Bai, Y. et al. 2022. Constitutional AI: Harmlessness from AI Feedback. *arXiv [cs.CL]*. Available at: http://arxiv.org/abs/2212.08073.

Brock, A., Donahue, J. and Simonyan, K. 2018. Large scale GAN training for high fidelity natural image synthesis. *arXiv [cs.LG]*. Available at: http://arxiv.org/abs/1809.11096.

Brown, T.B. et al. 2020. Language Models are Few-Shot Learners. *arXiv [cs.CL]*. Available at: http://arxiv.org/abs/2005.14165.

Chen, C.-H., Weng, M.-F., Jeng, S.-K. and Chuang, Y.-Y. 2008. Emotion-based music visualization using photos. In: *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 358–368. Available at: http://dx.doi.org/10.1007/978-3-540-77409-9_34.

Chen, L., Srivastava, S., Duan, Z. and Xu, C. 2017. Deep cross-modal audio-visual generation. *arXiv [cs.CV]*. Available at: http://arxiv.org/abs/1704.08292.

Cramer, A.L., Wu, H.-H., Salamon, J. and Bello, J.P. 2019. Look, listen, and learn more: Design choices for deep audio embeddings. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019/5/12-2019/5/17, IEEE. Available at: http://dx.doi.org/10.1109/icassp.2019.8682475.

Deng, L. 2012. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine* 29(6), pp. 141–142. Available at: http://dx.doi.org/10.1109/msp.2012.2211477.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. 2018. BERT: Pre-training of deep bidirectional Transformers for language understanding. *arXiv [cs.CL]*. Available at: http://arxiv.org/abs/1810.04805.

Dhariwal, P. and Nichol, A. 2021. Diffusion models beat GANs on image synthesis. *arXiv [cs.LG]*. Available at: http://arxiv.org/abs/2105.05233.

Drossos, K., Adavanne, S. and Virtanen, T. 2017. Automated audio captioning with recurrent neural networks. *arXiv [cs.SD]*. Available at: http://arxiv.org/abs/1706.10006.

Elman, J.L. 1990. Finding structure in time. *Cognitive science* 14(2), pp. 179–211. Available at: http://dx.doi.org/10.1207/s15516709cog1402_1.

Goodfellow, I.J. et al. 2014. Generative Adversarial Networks. *arXiv [stat.ML]*. Available at: http://arxiv.org/abs/1406.2661.

Gregor, K., Danihelka, I., Graves, A., Rezende, D.J. and Wierstra, D. 2015. DRAW: A recurrent neural network for image generation. *arXiv [cs.CV]*. Available at: http://arxiv.org/abs/1502.04623.

Hu, E.J. et al. 2021. LoRA: Low-Rank Adaptation of large language models. *arXiv [cs.CL]*. Available at: http://arxiv.org/abs/2106.09685.

Kingma, D.P. and Welling, M. 2013. Auto-Encoding Variational Bayes. *arXiv [stat.ML]*. Available at: http://arxiv.org/abs/1312.6114.

Li, J., Li, D., Savarese, S. and Hoi, S. 2023. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv [cs.CV]*. Available at: http://arxiv.org/abs/2301.12597.

Manco, I., Benetos, E., Quinton, E. and Fazekas, G. 2021. MusCaps: Generating Captions for Music Audio. *arXiv [cs.SD]*. Available at: http://arxiv.org/abs/2104.11984.

Mansimov, E., Parisotto, E., Ba, J.L. and Salakhutdinov, R. 2015. Generating images from captions with attention. *arXiv [cs.LG]*. Available at: http://arxiv.org/abs/1511.02793.

Mikolov, T., Chen, K., Corrado, G. and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv [cs.CL]*. Available at: http://arxiv.org/abs/1301.3781.

Nichol, A. et al. 2021. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv [cs.CV]*. Available at: http://arxiv.org/abs/2112.10741.

Oh, T.-H., Dekel, T., Kim, C., Mosseri, I., Freeman, W.T., Rubinstein, M. and Matusik, W. 2019. Speech2Face: Learning the face behind a voice. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019/6/15-2019/6/20, IEEE. Available at: https://openaccess.thecvf.com/content_CVPR_2019/papers/Oh_Speech2Face_Learning_the_Face_Behind_a_Voice_CVPR_2019_paper.pdf [Accessed: 25 July 2025].

Passalis, N. and Doropoulos, S. 2021. deepsing: Generating sentiment-aware visual stories using cross-modal music translation. *Expert systems with applications* 164(114059), p. 114059. Available at: http://dx.doi.org/10.1016/j.eswa.2020.114059.

Qiu, Y. and Kataoka, H. 2018. Image generation associated with music data. pp. 2510–2513. Available

at: https://openaccess.thecvf.com/content_cvpr_2018_workshops/papers/w49/Qiu_Image_Generation_Associated_CVPR_2018_paper.pdf [Accessed: 25 July 2025].

Radford, A. et al. 2021. Learning transferable visual models from natural language supervision. *arXiv [cs.CV]*. Available at: http://arxiv.org/abs/2103.00020.

Radford, A., Metz, L. and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv [cs.LG]*. Available at: http://arxiv.org/abs/1511.06434.

Ramesh, A. et al. 2021. Zero-shot text-to-image generation. *arXiv [cs.CV]*. Available at: http://arxiv.org/abs/2102.12092.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C. and Chen, M. 2022. Hierarchical text-conditional image generation with CLIP latents. *arXiv [cs.CV]*. Available at: http://arxiv.org/abs/2204.06125.

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B. and Lee, H. 2016. Generative adversarial text to image synthesis. *arXiv [cs.NE]*. Available at: http://arxiv.org/abs/1605.05396.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P. and Ommer, B. 2021. *High-Resolution Image Synthesis with Latent Diffusion Models*. Available at: http://arxiv.org/abs/2112.10752 [Accessed: 19 July 2025].

Rombach, R., Blattmann, A., Lorenz, D., Esser, P. and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022/6/18-2022/6/24, IEEE. Available at: http://dx.doi.org/10.1109/cvpr52688.2022.01042.

Ronneberger, O., Fischer, P. and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv [cs.CV]*. Available at: http://arxiv.org/abs/1505.04597.

Saharia, C. et al. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv [cs.CV]*. Available at: http://arxiv.org/abs/2205.11487.

Singh, J. 2023. Dataset of 65000+ Songs on Spotify(with images). Available at: https://www.kaggle.com/datasets/jashanjeetsinghhh/songs-dataset [Accessed: 28 July 2025].

Sohl-Dickstein, J., Weiss, E.A., Maheswaranathan, N. and Ganguli, S. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. *arXiv [cs.LG]*. Available at: http://arxiv.org/abs/1503.03585.

Soleymani, M., Aljanaki, A. and Yang, Y.-H. 2018. *DEAM: MediaEval database for emotional analysis in music*. Available at: https://cvml.unige.ch/databases/DEAM/manual.pdf [Accessed: 26 July 2025].

Sutskever, I., Vinyals, O. and Le, Q.V. 2014. Sequence to sequence learning with Neural Networks. *arXiv [cs.CL]*. Available at: http://arxiv.org/abs/1409.3215.

Turing, A.M. 1950. I.—computing machinery and intelligence. *Mind; a quarterly review of psychology and philosophy* LIX(236), pp. 433–460. Available at: https://dx.doi.org/10.1093/mind/LIX.236.433.

Vaswani, A. et al. 2017. Attention is all you need. *arXiv [cs.CL]*. Available at: http://arxiv.org/abs/1706.03762.

Wu, H.-H., Seetharaman, P., Kumar, K. and Bello, J.P. 2022. Wav2CLIP: Learning robust audio representations from CLIP. *arXiv [cs.SD]*. Available at: http://dx.doi.org/10.31219/osf.io/r2vwf.

Wu, T.-L. and Jeng, S.-K. 2008. Probabilistic estimation of a novel music emotion model. In: *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 487–497. Available at: http://dx.doi.org/10.1007/978-3-540-77409-9_46.

Yariv, G., Gat, I., Wolf, L., Adi, Y. and Schwartz, I. 2023. AudioToken: Adaptation of text-conditioned diffusion models for audio-to-image generation. *arXiv [cs.SD]*. Available at: http://arxiv.org/abs/2305.13050.

Zeng, M., Tan, X., Wang, R., Ju, Z., Qin, T. and Liu, T.-Y. 2021. MusicBERT: Symbolic music understanding with large-scale pre-training. *arXiv [cs.SD]*. Available at: http://arxiv.org/abs/2106.05630.

Appendix