A Multimodal Fusion of Fine Art, Emotion-Driven Captions, and Music.

Maria Alejandra Rivas Adames

Student ID: 2031033
Programme: MSci Computer Science

Supervisor: Dr Hyung Jin Chang

Word count: (10,184)
60 credits Masters

UNIVERSITY OF BIRMINGHAM

Abstract

The paper introduces an integrated system for infusing emotional captioning and music generation. The system aims to provide users with a novel platform for expressing and communicating emotions through visual and auditory mediums, leveraging state-of-the-art AI models. It explores innovative methodologies and multi-modal architecture for captioning, such as different convolution and recurrent neural networks. Through extensive testing and evaluation, the system demonstrates the ability to generate emotionally resonant captions for a wide range of images. On the other end of the system, the music generation components employ models such as AudioLDM, promising an avenue for generating semantic prompt-based music. Insights gained from the development process highlight the challenges and opportunities in integrating emotion into AI-generated content, underlining the way for future research and development in this domain.

Acknowledgements

I would like to express gratitude to my project supervisor, Dr Hyung Chang, whose guidance, and suggestions helped me through the most struggling times of the projects. The regular meetings and attentive ear provided invaluable navigation through the obstacles faced. I would like to extend my gratitude to my project inspector Dr Rishiraj Bhattacharyya, for their insightful comments and advice on how to write the report.

I am indebted to my supportive roommates, who patiently listened to the multiple complaints of “it just doesn’t work” only to realise I called the wrong variable a few moments later. Furthermore, I extend my gratitude to my friends and family for offering their time during evaluation and providing constructive feedback, plus nodding along to the barrage of jargon as I tried to explain where I was stuck at.

Finally, I would like to thank Aladdin Persson [1] on YouTube for their comprehensive playlist on how to use PyTorch.

Contents

1 INTRODUCTION 4
Motivation 4
Outline 4
2 RESEARCH 6
2.1 Similar Work 6
2.2 Image Recognition in Fine Art 6
2.3 Music Composition from Text 7
2.4 Gaps in the Current Literature 8
2.5 Conclusion 9
3 SYSTEM REQUIREMENTS 10
3.1 Model Training Requirements: 10
3.2 Training Infrastructure Requirements: 10
3.3 Functional Requirements: 10
3.4 Non-Functional Requirements: 11
4 DESIGN 13
4.1 Data Extraction 13
ArtEmis 13
MusicCaps 14
4.2 Architecture 14
Canvas to Emotion : Model A 14
Emotion to Music 15
Canvas to Music: Model B 17
4.3 User Interface Design 18
5 IMPLEMENTATION 20
5.1 Canvas to Emotion: Model A 20
Data Extraction 20
Data Management 21
Models 22
Training 23
5.2 Emotion to Music 24
MusicLM 24
AudioLDM 27
5.3 Application(UI) 29
6 TESTING AND EVALUATION 32
6.1 Canvas to Emotion: Model A 32
6.2 Application (UI) 36
7 PROJECT MANAGEMENT 37
Milestones 38
Setbacks 38
8 CONCLUSION 40
Impact 40
Future Steps 40
Final Statement 40
9 REFERENCES 41
10 APPENDIX 42
Instructions for Repository 42

1 Introduction

With Artificial Intelligence entering various domains, its presence in fine art, emotion analysis and music generation is increasing, therefore presenting new avenues for interdisciplinary investigation. This report delves into the convergence of these key elements through machine learning techniques.

Background literature suggests a promising frontier for machine learning, particularly in areas of image recognition with relevance to emotion and captivating audiences. This work seeks to explore further how these techniques are applied and utilised, in addition to recreating the results, specifically in areas of feature extraction and subjective analysis of these qualities. Essentially teaching an algorithm to be opinionated. Moreover, there is a great deal to explore within the interconnection of the fine art domain with the musical one. For this, the report will explore state-of-the-art algorithms for music generation, specifically with a focus on semantic understanding of captions. The intersection of these disciplines offers unique opportunities for creative expression and critical discourse, by combining the two, the aim is to blur the boundaries between these art forms.

Motivation

Stakeholders in this endeavour encompass individual entities, including artists, musicians, technologists, and educators. Artists and musicians stand to benefit from the exploration of these, with the aim to facilitate and inspire creative exploration. Technologists and enthusiasts of AI are motivated by the capabilities of the latest in machine learning, especially when finding a new way to contextualise machine learning in culture. Finally, educators have the potential to leverage AI to foster critical thinking skills among students.

The fine art to music systems initiatives aspire to be at the forefront of systems aiming to explore different forms of artistic development. Affording potential users a distinctive avenue for interacting with fine art through the conduit of music can not only provide an expanded avenue for experiencing art but also offer a platform for critical expression. The fusion of both aesthetics is bound to cause a diverse response in users: agreement, disagreement, or generating a new perspective on the artistic piece; this multifaceted interaction can enrich individual experience and foster critical thinking and a deeper contemplation of art and music, encouraging reflection and appreciation of the artistry. In essence, the integration of fine art and music serves as a catalyst for discourse and introspection thereby cultivating a more profound understanding of the aesthetics and cultural dimensions of art.

Broader societal impetus also includes inspiration in ongoing research endeavours within neuroscience. Contemporary studies [2], [3] explore the therapeutic potential of intertwining music and art to stimuli memory recognition and formation. Therefore, aligning our system with the interdisciplinary exploration of art, science, and human cognition.

Outline

The fundamental aim is to explore and understand the emergence of AI techniques into the realms of image recognition and analysis and music generation. Specifically, the objectives can be broken down into:

1. Training a system that can turn a painting into an emotion-focused caption.

2. Developing a system that can emotively caption an image and produce relevant music for it.
3. Reviewing existing literature and identifying gaps in research related to image recognition, emotion analysis, and music composition.
4. Developing a conceptual framework and architectural design for a system that synthesizes fine art, emotion analysis, and music generation.
5. Designing user interface components to facilitate intuitive interaction and customization for users interested in exploring the convergence of art and technology.

By addressing these objectives, this report aims to contribute to the ongoing discourse on the intersection of AI and the arts, providing insights into the potential applications and implications of integrating machine learning into artistic processes and experiences.

This report will be broken into the following sections. Section 2 will explore the literature available on these domains, together with an evaluation of each model explored. Section 3 will describe the system requirements and a breakdown of the aims of the project. It will separate the training and ML requirements from the Application ones. Section 4 explored the design of the system. The initial conceptualisation of how the backend of training and prediction will connect with each other and how this will then connect to the User Interface and Application presented to potential users. Section 5 will look at the implementation and explain any potential deviations from the design. Section 6 will look at the concluding effectiveness of the system through evaluation and testing.

2 Research

The coverage of image recognition, emotion analysis and translation to different sensory stimuli stands as a promising frontier in Machine Learning. This section aims to provide insight on the landscape between these interdisciplinary domains. In addition, it will show the significance as to how the merger lies not only in technological advancement, but also in the bridging of the two artistic realms itself.

The research will be broken down into: discussing the existing body of research in image recognition, specifically when focused on subjective qualities, the technologies around musical composition, how music models adjust to different prompts. Once this is understood, the underlying principles and challenges in the intersection of both will be discussed.

2.1 Similar Work

Pix2Pitch explores a similar idea directly going from image to music generation [4]. They use Generative Adversarial Networks. Specifically, they use the image representation of music (Mel-Spectrograms) and images to generate music. The researcher uses a wide range of models. Our aim deviates from theirs since for our system integrates a step in between the creation a caption and music generation. This is because it allows up to change the text for user customisation and personalisation.

2.2 Image Recognition in Fine Art

Wilber's BAM! [5] aims to provide a comprehensive database for artistic learning for machine learning. Stating that the image recognition often relies on photographic learning, where artists usually do not render images as it is presented in real life. Providing combinations and impossibilities. Therefore, a collection of contemporary art was gathered and tagged using a hybrid crowdsourcing and machine classification. For emotion attributes, a modified version of StyleNet was used. They trained this model to output a single attribute score using binary class-entropy loss. In this paper they asked a key question for this project "How well can object recognition models transfer to emotion and media classification?" They compared their trained StyleNet on BAM and a linear SVM on StyleNet features. Relevant to our task, their results showed that emotion detection and classification could be done from both models relatively successfully. With the fine-tuned StyleNet being the most successful. Overall, BAM! provides a valuable resource for exploring the relationships between different features and various attributes of a piece of art, specifically, understanding feature classification from imagery.

Mentioned above is the model StyleNet [6],the aim of this project is to produce stylistic captioning from images. Such as adding a romantic or humorous intent to the text output from an image. This model uses CNN and RNN for image captioning. Specifically, they introduce a factored LSTM model used to disentangle factual captioning from the stylistic one.

The proposed LSTM model introduces factorization to the input matrixed (W_x). The LSTM model uses shared matrices that capture the generic language generation process, shared across different styles. It then introduces a Style-Specific matrix $\{S\}$ different to each style (factual, romantic, humorous). Therefore, when captioning and generating the vocab for an image it can learn to caption factually and then weigh in style depending on the parameter specified.

A different approach is taken by 'Prose for a Painting' [7], from a created database they aimed to produce a Shakespeare prose from a painting. For feature extraction they implemented a 3-model parallel architecture of CNNs. More interestingly, joined it using a sequence-to-sequence model and two Discriminator Networks. One focused on assessing how well the generated text describes the input image, and the other evaluated the poetic quality of the generated text.. As for the captioning, they used a pre-trained word embeddings using a dictionary mapping between Shakespearean words and modern English words. Their results where promising with averaging results of 78% accuracy from 32 students surveyed.

Garcia [8] explored how different Image Encoders and Text Encoders performed. For feature extraction, VGG16, ResNet and RMAC (CNN models) was used. They then tested bag-of-words (BOW), multi-layer perceptron (MLP) and LSTM. They aimed to capture the semantic similarities between visual and textual representation by transforming these two into multi-modal space. The work proposes different models for this transformation, they will not be specified as it goes beyond the scope of this project. In summary, rather than captioning image they wanted to test how text features and image features can be correlated. The systems evaluation is compared to a human's results showing that the best performing multi-modal space vs human evaluation has ten points of difference. Concluding that "although there is still room for improvement, meaningful art representations are being obtained." This literature serves as a significant benchmark, illuminating key methodologies and insights that for the exploration of fine art to text to music synthesis.

Pivotally, Achlioptas's ArtEmis [9] utilises a combined philosophy for captioning images. For starters ArtEmis provides an essential dataset of fineart from WikiArt and human caption and classification from 9 emotional labels. In total the dataset contains 80k images and 450k attributes for each. Equal frequency of positive and negative emotions was collected with an overlap in 61% of the painting. They tested two approaches for captioning. What they refer to as Show-Attend-Tell (SAT), using a ResNet pretrained on ImageNet, and Meshed-memory transformers (M2): a recent line of work that employs top-down, bottom-up meshed-memory transformers. It uses separate object-bound-box detection with transformer units rather than RNN. Both approaches were tested using Neural Speakers Evaluation Metrics (such as BLUE 1 to 4, METEOR) and human Turing Test. Both approaches performed similarly for the neural speaker's evaluation metrics with SAT having a higher fraction of similes in its generated captions compared to M2. The Turing test indicating that, for half of the instances, the SAT's model generated explanations comparable or similar to explanations that a human might provide. Achlioptas's work serves as a robust reference, informing our approach to emotion-laden caption generation.

2.3 Music Composition from Text

There are a few notable pieces of literature examining turning text into music. Specifically, systems that can semantically understand the meaning of a prompt and turn that into text are of interest. MusicLM [10] proposes using a model that expands on AudioLM [11]. To break down how each works, AudioLM will be examined first.

AudioLM is a framework designed for high-quality audio generation with long-term coherent structure. It borrows two tokenizer technologies: a *semantic tokenizer* such as wev-2-Bert[12] and an *audio codec* such as SoundStream [13]. SoundStream adopts a convolutional encoder, which processes the input waveform to generate a sequence of embeddings. Wev-2-Bert, also a self-supervised learning model produces semantic tokens by mapping the audio waveform to a rich set of linguistic features, capturing aspects like phonetics, syntax, and semantics.

There is an inherit loss of sampling rate when it comes to both tokenizers. To conserve this is creates a hybrid tokenizer, the acoustic tokens capture detailed audio information, while the semantic tokens contribute to long-term structural coherence.

The architecture for AudioLM's audio generation process is broken down into multiple stages. Firstly, semantic modelling decoder-only transformer is used to capture the autoregressive prediction of semantic tokens. It focuses on long-term temporal structures in the audio, focusing on linguistic content, and other semantic features. Secondly coarse modelling, another decoder only transformer, it uses the SoundStream tokens and semantic tokens from the semantic tokens from Wev-2-Bert to predict acoustic tokens conditioned on both the semantic tokens and previously generated tokens, meaning it on preserving key acoustic properties (such as speaker identity and recording conditions). Finally, the fine modelling: this is used to refine audio quality such as removing artifacts and noise.

MusicLM [10]expands on this wanting to achieve text conditioning for the purpose of generating music. Using their library for training, MusicCaps, it builds upon AudioLM's framework. The key component introduced in MuLan, a joint music-text embedding model. MuLan consists of two embedding towers, music, and text. These towers map music and text representations to a shared embedding space using a contrastive learning approach (the idea being that will learn to map text and music pairings while separating unrelated pairs). This allows the model to learn the connection between music and text. The text embedding network employs BERT model, which is pre-trained. The audio tower uses ResNet-50 to extract embeddings from music clips.

For training MuLan pairs a collection of descriptors to music, taken from MusicCaps. Importantly, MuLan imposes only weak requirements on the quality of its training data. The researchers evaluated their results as successful, scores in FAD metrics (FADVGG) being better compared to baseline models. Finally, in a human listening test, MusicLM is clearly preferred over both baselines in pair-wise comparisons.

AudioLDM [14] uses a similar principal to MuLan. The key difference is the Latent Diffusion Model (LDM) and the specification and use of Contrastive Language-Audio Pretraining (CLAP) [15]. The system is pre-trained, unlike with the MusicLM model seen above. Once the model is trained, we introduce the other key piece of technology, Latent Diffusion Models. This is a probabilistic generative model that estimate the conditional data distribution based on a text description. The model can be split into two processes. The forward process to transform data distribution into a standard Gaussian distribution. The reverse process to gradually generate data sampled from the noise. To enhance the training of LDMs, mix-up augmentation is performed on audio-only signals. Using similar evaluation methods as MusicLM, AudioLDM also shows promising results. With performance The Overall Quality Score Relevance to the Input Text scores around 64.

2.4 Gaps in the Current Literature

White exiting works like Pix2Pitch offer an image to music creation, there is a gap in customisation and personalization. The proposed project emphasizes the importance of the text output as an intermediate step. Allowing users to change the prompt to create a varied generated music. A lot of the captioning models focus on a image recognition, however there is potential gap in the exploration of fine art emotion analysis, similarly to the ArtEmis project. Moreover, some captioning tools like 'Prose for a Painting' and 'StyleNet' focus on a stylistic approach to captioning, but do not link it to another artistic medium such as music.

The introduction of text as an intermediate step allows us to expand on the exploration of emotion latent prompts and customisation when generating music. Building upon the stylistic and emotion ridden caption through a multi-modal system can provide a more comprehensive understanding of the relationships between different artistic dimensions. The proposed project therefore aims to integrate the theories from affective computing and semantic music representation.

2.5 Conclusion

The existing body of research has used different approaches, methodologies, and models for image recognition, some specifies use in fine art, with variation in stylistic captioning. The most promising aspect to investigate, ArtEmis and MusicLM continue significant building blocks to the creation of the project. Specifically, the contrastive exploration or parallel CNNs for image captioning, using a Show-Attend-Tell approach, like the one introduces by Achlioptas's and Meshed-memory transformers, allowing for fine-tune synthesis of fineart and emotional captioning. As for the music generation, AudioLDM's process shows promising results with a reasonable expectation of trainability and integration to the system.

3 System Requirements

Given how there is a joint Application and Machine Learning outcome for the system, it was thought important to provide clarity in development. Covering both the model training and the user interactivity with functional and non-functional requirements helps set a blueprint for testing and validation. Moreover, it is important to understanding the roadmap for a system that not only recognizes and interprets fine art, but bit is also able to craft a musical tune out of it. Therefore, setting guiding parameters, ensuring the methodical and successful implementation of the envisioned system.

3.1 Model Training Requirements:

1. Image Recognition and Emotion Analysis Models:
    - Fine-tune the caption model on the curated fine art dataset for improved accuracy in emotion recognition.
    - Train a separate emotion analysis model (e.g., using deep neural networks) on the annotated emotional cues dataset.
2. Music Generation Integration:
    - Integrate the trained emotion analysis model using an accelerator and fine-tune the joint system.
    - Experiment with different configurations and parameters to optimize the interaction between textual input and musical output.

3.2 Training Infrastructure Requirements:

1. Hardware:
    - Ensure access to powerful hardware, such as GPUs or TPUs, to expedite the training process for deep learning models.
2. Software:
    - Utilize deep learning framework PyTorch for model training.
    - Implement version control systems for tracking model versions and facilitating collaboration.
3. Data Pre-processing:
    - Implement data pre-processing pipelines to clean, augment, and prepare the fine art image, text, and audio datasets for training.
    - Consider techniques such as normalization, data augmentation, and embedding extraction.

3.3 Functional Requirements:

1. Image Recognition and Emotion Analysis - Model A:
    - It **must** include a fine-tuned emotion analysis model based on a diverse dataset of emotional cues in fine art.
    - The system **must** be able to produce emotion caption from a range of art styles.
    - The system **must** be able to produce emption regardless of fine-art's subject matter.
    - The system **must** be able to produce emotion-based captions. This means include adjective subjective qualities.
    - The **must** be able to work with unseen data from the database.
2. Text to Music Translation – Model B:
    - The system **must** be able to interpret music from a text.

    - The system **must** produce sufficient variance from one text input to another.
    - Users **must** have the ability to customize and fine-tune the musical output based on the textual input through the user interface.
    - It **should** map extracted emotions to musical elements, including tempo, key, instrumentation, and mood.
3. User Customization:
    - The user interface **must** be intuitive and user-friendly, allowing users to modify recognized emotions and textual captions easily.
    - Customization options **should** include sliders, buttons, or interactive elements to adjust musical parameters and styles.
    - Real-time feedback on the impact of user modifications on the music output **should** be provided.
    - The user **must** be able to input their own images to the model.
4. Audio Playback and User Interaction:
    - Users **should** have the capability to audibly experience the system's output.
    - The system **should** provide a user-friendly interface that allows users to play and pause the generated musical composition.
    - The system **could** implement features enabling users to easily locate and access the extracted audio file associated with their customized compositions.

3.4 Non-Functional Requirements:

1. Performance:
    - The system **should** process image recognition and emotion analysis in real-time, providing prompt feedback to users.
    - Music composition **should** be efficient, with low latency between user input and the generation of the final musical output.
2. Scalability:
    - Scalability considerations **should** be made for potential future expansions, such as integrating additional pretrained models or datasets.
3. Reliability:
    - The system **must** be reliable and stable, minimizing the occurrence of errors or crashes during user interactions.
    - Robust error handling and recovery mechanisms should be implemented.
4. Usability:
    - The system **must** have clear documentation and onboarding materials should be provided.
    - The user interface **should** be designed for ease of use, catering to users with varying levels of technical proficiency.
5. Compatibility:
    - The system **should** be compatible with a range of devices and browsers to ensure accessibility for a broad user base.
    - Compatibility with different screen sizes **should** be considered.
    - The system **could** implement a system that seamlessly switches to CPU usage if a dedicated MPS (Metal Performance Shaders) is not available, ensuring consistent performance across diverse hardware configurations.
6. Virtual Environment Integration:
    - The system **should** include a *requirements.txt* file to specify dependencies for creating a virtual environment (venv).
7. Maintainability:

    - The codebase **should** be sufficiently documented and maintainable to facilitate future updates, enhancements, and bug fixes.
    - The system **should** be design for regular maintenance and updates to pretrained models and datasets should be planned.
8. Interoperability:
    - The system **must** support standard data exchange formats, allowing for seamless integration.
    - The system **must** utilise common protocol formats such as JSON, CSV and pickle (pkl), enabling effective communication between platforms.
9. Memory:
    - The system **must** be able to run within operational efficiency by running on a standard CPU (without necessitating GPU or TPU dependencies)

4 Design

From the literature explored and the requirements laid out, an intuition for the design was crafted. This section delineates the architectural framework for both the training and model prediction, as well as user interface design envisioned for the proposed system. The design aims to create a division from the emotion captioning and the music generation. This was done to ensure the accuracy of one model would not hinge the development of the other, furthermore allowing for parallel training and implementation. Building upon the research above, the web scrapping for both databases used in the project (ArtEmis[9] and MusicCaps[10]).

4.1 Data Extraction

4.1.1 ArtEmis

The ArtEmis Database consisted of a csv file with over 450k entries. The paper provided mentioned that the art was taken from WikiArt, therefore based on the layout of the data provided an idea of using a regex composition from for the URL was thought of. The assumption was that the csv file had a naming convention consistent to that on the WikiArt URL path. The location of the URL would be written on the csv file and then a separate function would find the suitable image from the page.

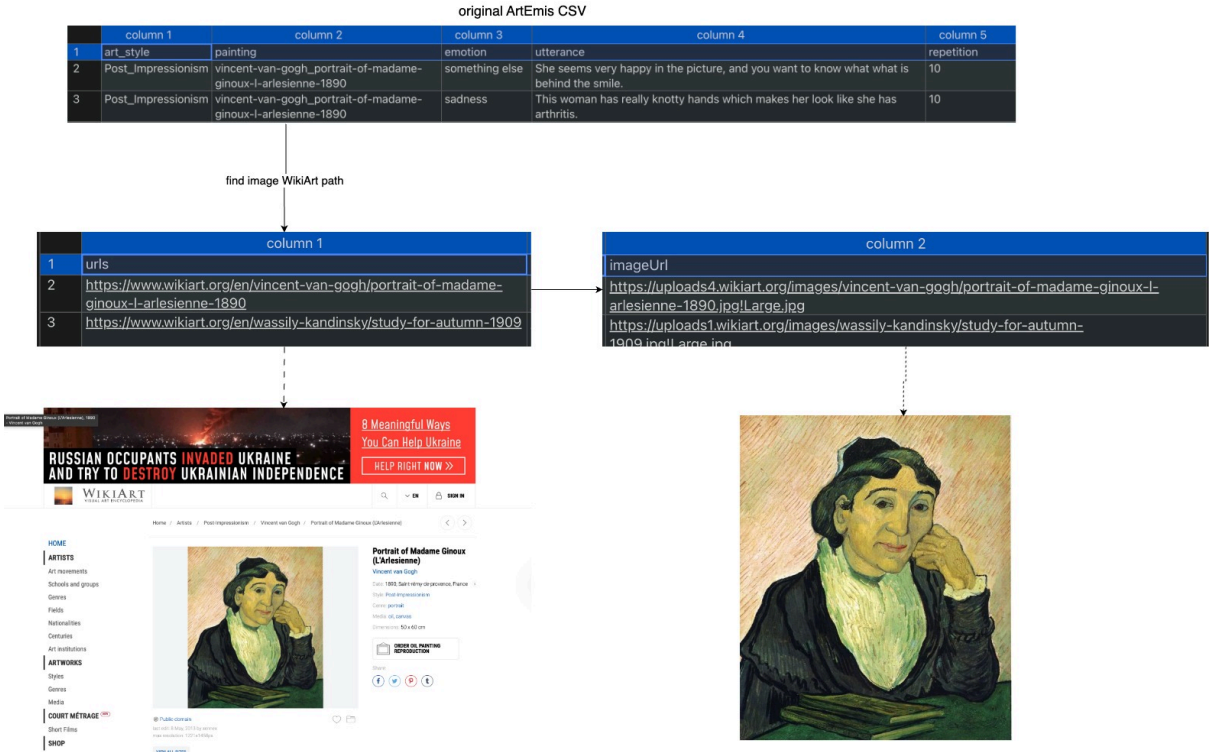


Figure 1. Diagram showing the design workflow for painting/canvas extraction.

For storage in the system, the images would then be resized so none would exceed the maximum size of 250 x 250. The images would then be stored using the <<painting>>.jpg template. It is important for the image naming convention to stay consistent with the csv file.

4.1.2 MusicCaps

A similar approach would be taken for the music extraction. The audio segment would we extracted and downloaded using 32k bit rate. Music caps uses audio clips of 10 seconds so this would keep the sizing consistent and easier to process – where each audio clip would be saved as <<ytid>>.wav – facilitates retrieval and processing of the audio when training. A critical challenge for this audio extraction would be in bypassing restricted content, whether due to age-restriction or based on country-dependent legislations. To address this, YouTube access would be accessed through Google's API to create a key that would help is overcome these limitations.

	column 1	column 2	column 3	column 4	column 5
1	ytid	start_s	end_s	aspect_list	caption
2	-0Gj8-vB1q4	30	40	['low quality', 'sustained strings melody', 'soft female vocal', 'mellow piano melody', 'sad', 'soulful', 'ballad']	The low quality recording features a ballad song that contains sustained strings, mellow piano melody and soft female

Figure 2. Table sample of how the MusicCaps data is presented.

4.2 Architecture

4.2.1 Canvas to Emotion : Model A

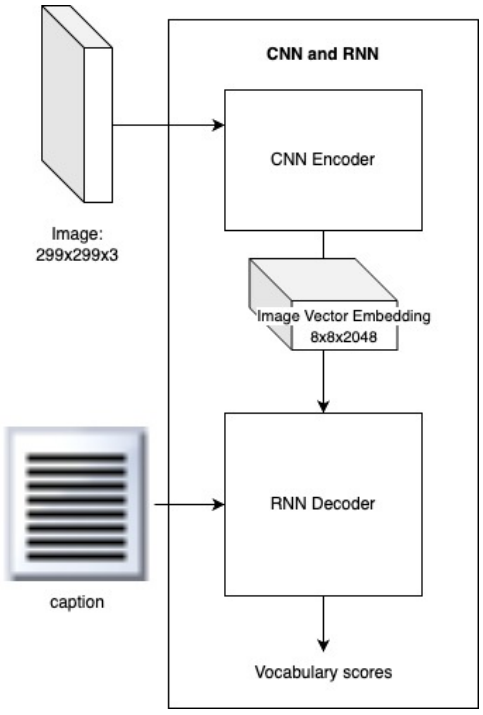


Figure 3. High level view of the Image Captioning Multi-Model System

The system proposed for the initial section of Art to Emotion entails the development of two primary models. One model will be dedicated to generating image embeddings, it's task therefore will be feature extraction of the input images. The other model will focus on

constructing the vocabulary to facilitate the conversion of image embeddings to textual descriptions.

The CNN Encoder will adhere to ArtEmis' 'Show-Attend-Tell' framework for extraction of emotional attributes from fineart. Whereas ArtEmis uses ResNet, our system will use inception_v3[16], this choice is informed by several factors: Inception_v3 offers a well-balanced trade-off between computational complexity and performance. The project is limited by hardware performance and a time schedule, therefore being able to produce sufficient results that are accurate enough was deemed as more important at this stage of the project. Moreover, its inception modules facilitate the extraction of multi-scale features, allowing for a comprehensive representation of fine art images.

When extracting features, a pretrained model will be used. It will be retrained with our database to finetune emotional language and interpretation of art. A challenge would be to have the image be able to interpret accurate emotion when the image has no subject. For example, and abstract piece being able to evoke sentiment from colour, shape, and texture rather than the expression of a portrait.

The second part of the captioning system will be to turn the vector image embeddings into Vocabulary Scores, building the RNN Decoder, essentially the model's predictions for the next word in the caption sequence at each time step. For this LSTM will be used, previous literature has proved it to be efficient and the most suitable for capturing he context and semantics of the image and the caption being generated. Finally, after the LSTM layer, to map the hidden states to vocabulary scores a standard Linear layer will be used. This will allow the model to output a distribution over the vocabulary, assigning probabilities to each word in the sequence.

To facilitate the translation from the CNN Encoder to the RNN Decoder a third model will be used. It will combine the feature extraction of the CNN Encoder and vocab prediction of the RNN Decoder. In addition, this model will have a rudimentary captioning feature for after the model is trained. This function will iterate for each word generated on an image.

4.2.2 Emotion to Music

The initial blueprint for the music creation would have follow closely MusicLM's framework. As the project progressed, however, a significant challenge arose in the implementation phase. MusicLM's comprehensive framework, extensive documentation and active git site initially made it an appealing choice, however, the obstacles faced (discussed in the implementation section) warranted a pivot to another framework: AudioLDM. Both designs will be elucidated below, providing clarity for future discussion on implementation.

4.2.2.1 MusicLM

The design for MusicLM would not deviate much further from what was proposed in the literature[10]. Some key differences will be applied so that it can be trained using Google Collab, as it provides a wider selection of GPU access and monitoring.

The system will use SoundStream for audio codec and Wav2Vec, specifically pretrained 'hubert_base_ls960' [17] for the Semantic Tokenizer. SoundStream will be trained first, using the audio provided by MusicCaps. Subsequently, MuLan will be trained using text and audio pairings. From that the quantizer will be applied to train each transformer.

By integrating these components and methodologies, the MusicLM framework aims to streamline the training process while maximizing computational resources for optimal model performance and efficacy. The predicted outcome is for SoundStream and MuLan to take the most substantial amount of time to train.

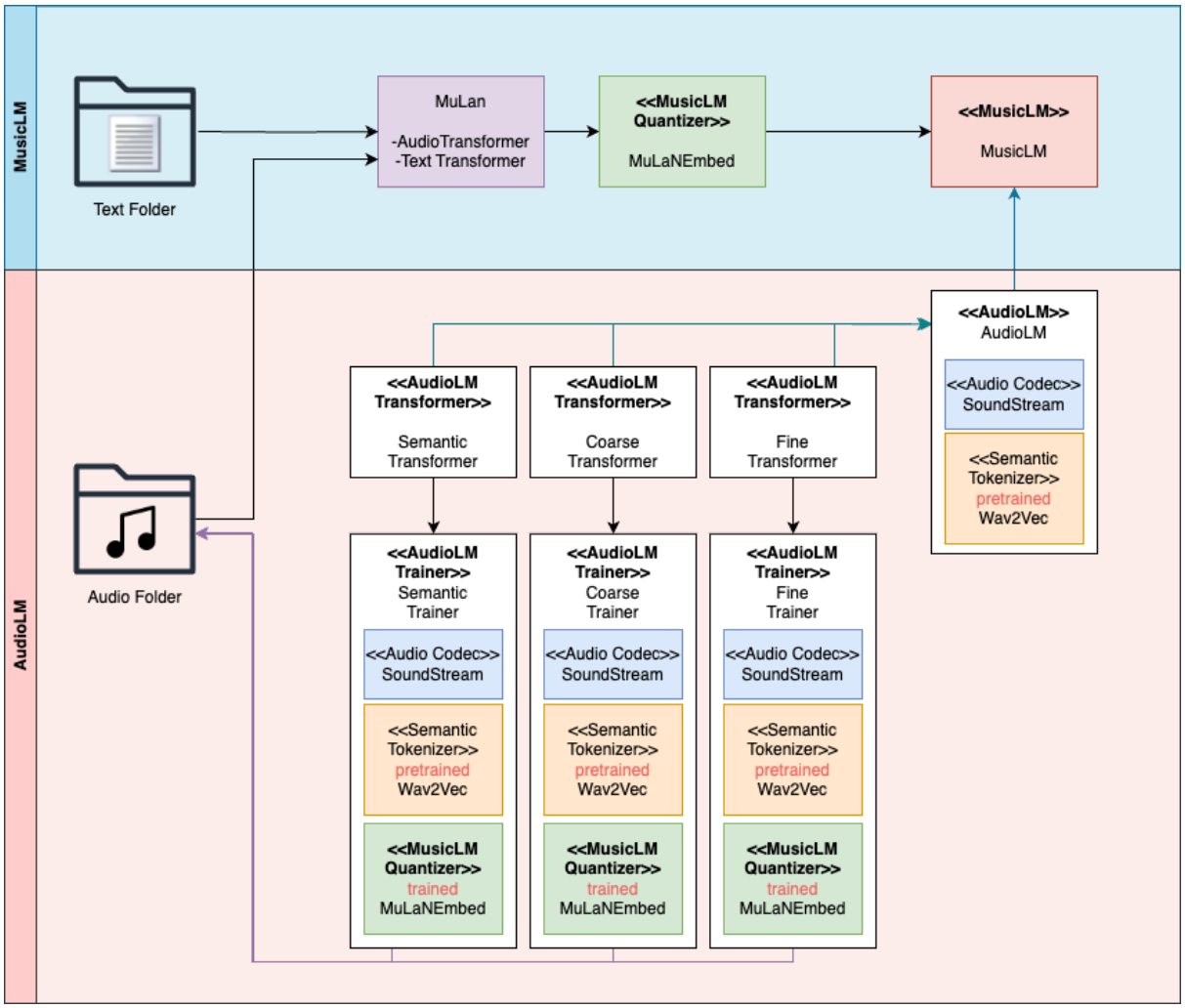


Figure 4. MusicLM and AudioLM merging multi-modal system. Information and transformer overview sharing modalities for training.

4.2.2.2 AudioLDM

Unlike MusicLM, AudioLDM does not necessitate training as it uses Contrastive Language-Audio Pretraining (CLAP), a pretrained model that enhances the semantic understanding of a prompt for Audio making. Leveraging this advantage, the design focus changes to focusing on optimization rather than training. Accordingly, an accelerator will be integrated, this will be complemented by a pipeline structure, leveraging the robust infrastructure provided by Hugging Face. Together these components ensure streamlined processing from performance in translating textual prompts into expressive musical compositions.

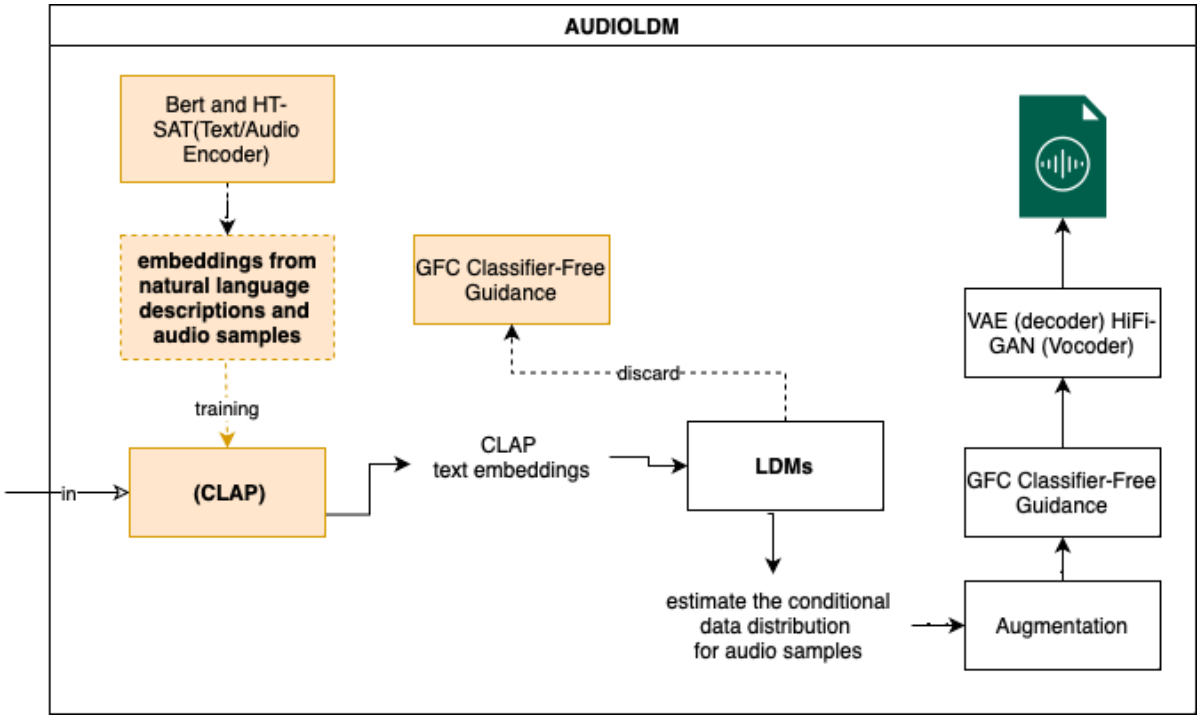


Figure 5. Overview summary of how AudioLDM works and LDM plus CLAP technologies.

4.2.3 Canvas to Music: Model B

Under the assumption that all models and trained an operational, the system will need to transition from the backend prediction models to the user interface, bridging the gap between complex algorithms and user interaction. To ease loading time, an initial cost will be taken at the beginning of the program rather than during its operation. This means most databases, pipelines and models will be loaded before the user gets a chance to interact with the program. For this we must design an architecture that can store and update depending on later user input (without too much time complexity)

The main classes involved in this process are as follows:

1. **mainWindow**: Representing the main window of the application. Pivotally it coordinates the interaction between the different screens and UI components. Such as buttons sliders and class specific widgets like the captioning_UI, imaging, and playback controls.
2. **loadingScreen**: This class provides visual feedback to the user during loading and processing tasks. It communicates with the Executive class to trigger and monitor the progress of backend tasks. It will also load each step and database involved in the overarching system. Importantly it should provide accurate visual representation and labelling of each step of the loading progress.
3. **CaptionUI**: Responsible for displaying and managing the user interface related to captioning, including model selection, text input, and output visualization.
4. **ImageUI**: Handles the user interface elements related to image processing, such as image upload, display, and interaction.

5. **Playback**: Manages audio playback functionality, including controls for starting, stopping, and adjusting volume. Also handles the music generation by calling the Executive class to generate music.
6. **Executive**: The central orchestrator of the system, the Executive class manages the flow of operations from image processing to music generation. It stores the audio pipeline and accesses the trained models for captioning and music generation. Importantly, the Executive class links the captioning process to music generation by passing the generated text to the music generation pipeline. It also stores the current CNN2RNN model, along with image, audio requirements, database for vocabulary, and captioning data.

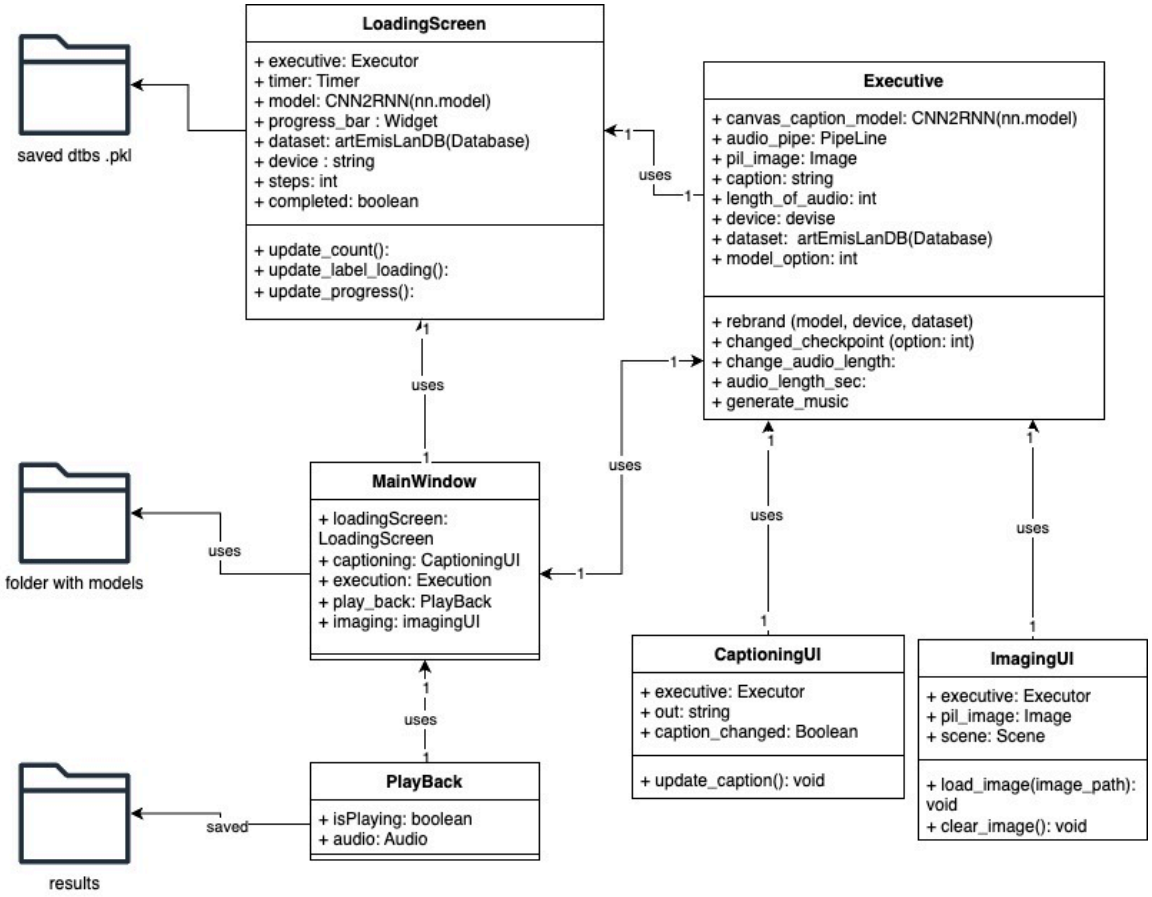


Figure 6. Class Diagram of the Visualization module integration with Model retrieval and Captioning. Together with communication with system file databases.

4.3 User Interface Design

The user interface follows some key ideologies to ease use and visualisation of the system: Simplicity, consistency, feedback, and hierarchy. Having a flow of control and purpose from one section of the window to another will enable to distinct each task of the Emotion to Music process.

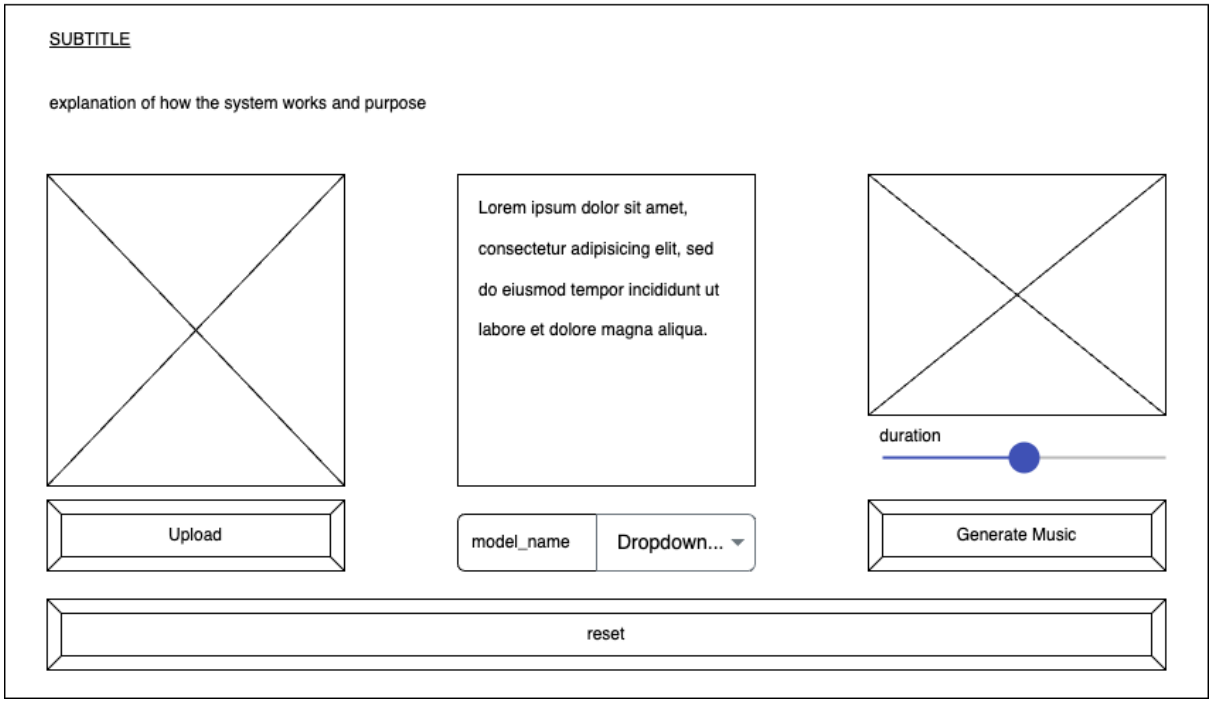


Figure 7. Wireframe of the UI Main Window

Specifically, the window's body will be split into three columns, each serving a specific purpose. The left column will be dedicated to the image-related function, this will include uploads, and visualisation of the images on the UI. The middle column will focus on the captioning: model selection, text visualisation and generation. To make it intuitive, the same box for music generation can be manually edited and written over, this means that the user need only understand how a single box works for customisation when it comes to the next column. The right column is allocated for music generation functionalities, this includes the button to generate music, some playback control, and a slider to select music generation duration. Finally, a reset button should be made available to clear the text and images displayed. Figure 7 displays the wireframe for the system.

Importantly there should be an element of feedback for every interactive element, specifically a visual response to user's touch and manipulation. Importantly, as the music generation has the potential to last for a long time before anything is on display, loading screens and animations will be used to let the user know that work is being done.

5 Implementation

5.1 Canvas to Emotion: Model A

5.1.1 Data Extraction

Downloading an image was split into two tasks. The first was to find a web URL that contained the image (preferably from WikiArt). Figure 8 shows the code snippet that does the following:

- For each row, if the 'urls' column is empty, it constructs a search query using the artwork details and 'wikiart' keyword.
- It performs a Google search for the constructed query and extracts the first URL that seems relevant to WikiArt.
- The URL is then stored in the 'urls' column of the Data Frame.
- The Data Frame is periodically saved to a CSV file named 'df_google.csv' during the extraction process.

```
except Exception:
    print(f"error name split {painting}")
try:
    from googlesearch import search
except ImportError:
    print("No module named 'google' found")

result_string = ' '.join(details)
query = f"{result_string} wikiart"
```

Figure 8. Code snippet of the URL extraction

The reason for saving URLs as CSV rather than streamlining the process was because, to not block the IP requests, there where pauses between each request. This meat that the download could take a few hours. Documenting which entries had been recorded with URLs periodically was essential since the code may unexpectantly terminate during the hours long process of finding each URL.

```
req = Request(url, headers={'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 '
                                          '(KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'})
htmldata = urlopen(req, timeout=2000)
soup = BeautifulSoup(htmldata, features: 'html.parser')
green_flags = ['Large', '/images/', artist, title, '.jpg', '.JPG']
red_flags = ['FRAME', 'banner_top', 'ARTIST', 'placeholder']
for idx, item in enumerate(soup.find_all('img')):
    elem = item['src']
    if not any(flag in elem for flag in red_flags):
        return elem
```

Figure 9. Code snippet of the Image URL extraction using Beautiful Soup Library

The second task for the data extraction was finding the Images' URL. Meaning something ending with an image extension that could be directly downloaded. Using Beautiful Soup, the 'img' tags from the HTML could be filtered. To evaluate whether the image was the targeted painting and not, for example a banner. For this a green-flag/red-flag system was used. These where again stored to CSV.

5.1.2 Data Management

To facilitate the training of out captioning model, a custom Database and Data Loader where developed utilizing PyTorch's library. The custom databased, named **artEmisLanguageDatabase,** plays a key role in accessing the training data and, more importantly it also incorporated the Vocabulary class, essential for the RNN.

1. **artEmisLanguageDatabase:** key class in charge of data storage, retrieval, and loading. As mentioned in the design, the goal was to leverage a more storage costly approach in order to accelerate the future application's time complexity.
    a. **Initialization:** The database initializes by parsing relevant information such as the image directory, the csv file (handled with panda) containing each textual description of the images and other parameters such as frequency threshold for vocabulary construction. It also takes in the transform composition for the images processed.
    b. Data Loading and Saving: Saves/Loads the database and build vocab into a pickle database in the system.
    c. Vocabulary Building: The database constructs a vocabulary based on the human captions provided. It is essential for numericalizing the text into a induces sequence that can be understood by the machine.
    d. Data Retrieval: When an index of the database is called (__getitem__) the class retrieves the appropriate .jpeg and applies any necessary transformation to it. The standard transformation for image processing in training consisted of the parameters shown in figure 10.

```
transform = transforms.Compose(
    [
        transforms.Resize((356, 356)),
        transforms.RandomCrop((299, 299)),
        transforms.ToTensor(),
        transforms.Normalize( mean: (0.5, 0.5, 0.5),  std: (0.5, 0.5, 0.5)),
    ]
)
```

Figure 10. Image default transform parameters and composition.

Importantly for the retrieval of an item, is the textual caption being returned as numericalized using the vocab construction of earlier. Those involved tokenizing the caption to individual words, replacing them with the corresponding indices and adding the starting and ending tokens "<SOS>" and "<EOS>".

2. **Vocabulary:** This class is responsible for managing the vocab, indexing and tokenizing of the dataset.
    a. **Initialization:** Initialises with special tokens ("<PAD>", "<SOS>", "<EOS>", "<UNK>") and a specified frequency threshold.

In addition, it initialises the String to Index (stoi) and Index to String (itos) dictionaries. These allow for efficient conversion of words into their numerical representations during text processing.

b. **Tokenization**: It includes a tokenizer (tokenizer_eng) utilizing NLTK's. NLTK's **word_tokenize** function provides a mechanism to tokenize textual data into individual words or tokens. This means that the function splits each sentence into words, with spaces, punctuation, and tabs as boundaries. In addition, it handles special cases such as contractions. The output therefore is a list where each element represents a tokenized word.
c. **Vocabulary Building**: The class constructs the vocabulary based on frequency of each for in the encompassing dataset. Words below the frequency threshold are replaced with "<UNK>" token. This helps manage vocab size and out-of-vocabulary words.
d. **Numericalization**: For each token, we check if it exists in the **stoi** dictionary (mapping words to indices). Corresponding tokens are appended, if they exist, to stoi. Otherwise, they will be added to "<UNK>". Finally, we return the **numericalized_sequence**, which contains the numerical indices representing the input text.

3. **MyCollate:** For batched training, a dataloader must be used. For this, a Collate class must be used for the dataloader to process each indivisual sample. When called, the collate sepparates images and targets (captions) from each batch. Images are then stacked into a flat tensor to form a batch of images. Teargets are passes using pad_sequence to ensire uniform length within the brach.
4. **DataLoader:**
    a. Parameters:
        - **batch_size**: Number of samples per batch.
        - **num_workers**: Number of worker processes for data loading.
        - **shuffle**: Whether to shuffle the data.
        - **pin_memory**: Whether to use pinned memory for faster GPU transfer.
        - **dataset**: Optionally, an existing dataset instance can be provided.

5.1.3 Models

The system uses nn.Module, leveraging PyTorch's neural network functionalities, specifically, as discussed in the design section: The captioning multi-modal system was broken down into three key elements:

5.1.3.1 Caption Encoder CNN

The model is designed using a pretrained Inception V3 architecture. To adapt it to retraining for the purpose of our system the last 3 layers where removed. Firstly, understanding the backbone of Inception_V3 is necessary. By default, the pretrained model includes auxiliary classifiers, which are used during training but not for feature extraction. The first layers of inception_v3 are used primarily for image processing and extraction of low-level features through convolutional and pooling operations. These helps identify basic patters and structures the image has, such as edges, texture, and shape. We keep these layers as these low-level features serve as building blocks for higher-level features extracted in deeper layers of the network. From the inception_v3, what we want to do is change the output. For this we need only change the Fully Connected Layer (fc). FC layer is replaced with a new Linear

Layer, we initialise it with a new embedded size, allowing to change our database's new classification scores.

The forward pass is now defined in the system. The image (input) is passed through the Inception_V3 backbone(feature extraction), from then on, the output features are passed through a ReLu activation function and a dropout layer to prevent overfitting and improve generalisation. The dropout layer randomly zeros some input elements, this will help is better anticipate unseen data and generalise it.

5.1.3.2 Caption Decoder RNN

For the RNN, we aim to create a sequence-to-sequence model for captioning. The key elements are: Embedding Layer, LSTM Layer and Linear Layer.

- The embedding later converts word indices (the construction of these explored above) and turns them into embedding vectors. This will be fine-tuned during training to capture the semantic relationship between words.
- The LSTM Layer is responsible for the embedded input sequence. It takes the embeddings from above and hidden states to compute the next probable state in the sequence.
- The Linear Layer maps the LSTM outputs into a vocabulary space, it takes the LSTM outputs and predicts the probability distribution over the vocab for each time step. Allowing the mode to predict the nest word in the sequence.

The same Dropout layer is applied for the same reasoning as the Encoder CNN.

5.1.3.3 Caption CNN to RNN

The final implementation for the captioning model is a class that streamlines the above models into one. As shown in figure 11 these are combines to create a caption output from an image input.

```
class CaptionCNN2RNN(nn.Module):
function __init__():
        self.encoderCNN = CaptionEncoderCNN()
        self.decoderRNN = CaptionDecoderRNN()
function forward(self, images, captions):
        features = self.encoderCNN(images)
        outputs = self.decoderRNN(features, captions)
        return outputs
```

Figure 11. Pseudocode for integration of CNN and RNN into a single model.

For future caption printout and prediction, this class also contains a caption image function. This method iteratively predicts words until either the maximum caption length is reaches or "<EOS>". The predicted indices are converted into words using the vocabulary (**itos** dictionary). Gradient computation is disabled during caption generation to speed up inference and save memory.

5.1.4 Training

With the images downloaded using the design in section 12 and the models set up, the training requires some key ideas:

- **model**: The captioning model initialized with the specified hyperparameters. It consists of an encoder CNN and a decoder RNN.
- **criterion**: For seq-to-seq modelling, the loss function is crucial. This will quantify the discrepancy between the predicted sequence and the ground truth, providing feedback to the model during training. Cross Entropy Loss us used with padding tokens ignored.
- **optimizer**: Adam optimizer is utilized for its adaptive learning rate; this will update the model parameters during training.
- We iterate the model after 10 epochs initially. Since the feature extraction is pretrained there is less needed to run it multiple times.
  As shown in figure 12. the optimizer is zeroed to clear any previously accumulated gradients. Gradients are computed using backpropagation which calculates the loss with respect to the model parameters.

```
iterate epoch:
        save_model()
        for imgs_batch, captions_batch) in train_data_loader:
            outputs = model(imgs_batch, captions_batch)
            loss = criterion(outputs, captions_batch)
            optimizer.zero_grad()
            accelerator.backward(loss)
            optimizer.step()
```

Figure 12. Pseudocode for the training algorithm.

After the 10th epoch was trained, the model was saved and used for the first evaluation. Another model was produced by changing the learning rate, vocab_frequency threshold and dropout rate. These will be discussed further in section 6.

5.2 Emotion to Music

5.2.1 MusicLM

Delineated in Section 5, it is described the implementation and methodologies when training models in MusicLM and AudioLM. The system followed closely the researchers' architecture of training.

5.2.1.1 SoundStream

Training SoundStream has been extensively researched and attempted, following closely the advice in GitHub, the parameters in figure 13 where chosen. The reason behind them is because:

```
soundstream = SoundStream(
    target_sample_hz = 24000,
    codebook_size = 1024,
    channels = 32,
    strides=(3, 4, 5, 8),
    rq_num_quantizers = 8,
    rq_groups = 2,
    multi_spectral_recon_loss_weight = 1e-2,
    adversarial_loss_weight = 1,
    feature_loss_weight = 100,
)
```

Figure 13. SoundStream model parameters.

- Strides of (3, 4, 5, 8) were selected to enable effective down-sampling and feature extraction at multiple scales.
- multi-channel processing, this focuses on the model's capacity to capture intricate audio details.
- Both rq parameters (residual quantification) manages the efficient compression and reconstruction of audio signals.

Using the model above and the MusicCaps audio library of music, for training, in total 150k epoch where iterated through. Total loss stabilized around 120k iterations to 1.25. After that, changing learning rates didn't affect much the loss. Moreover, there where time constraints and the Audio Codec provided sufficient results.

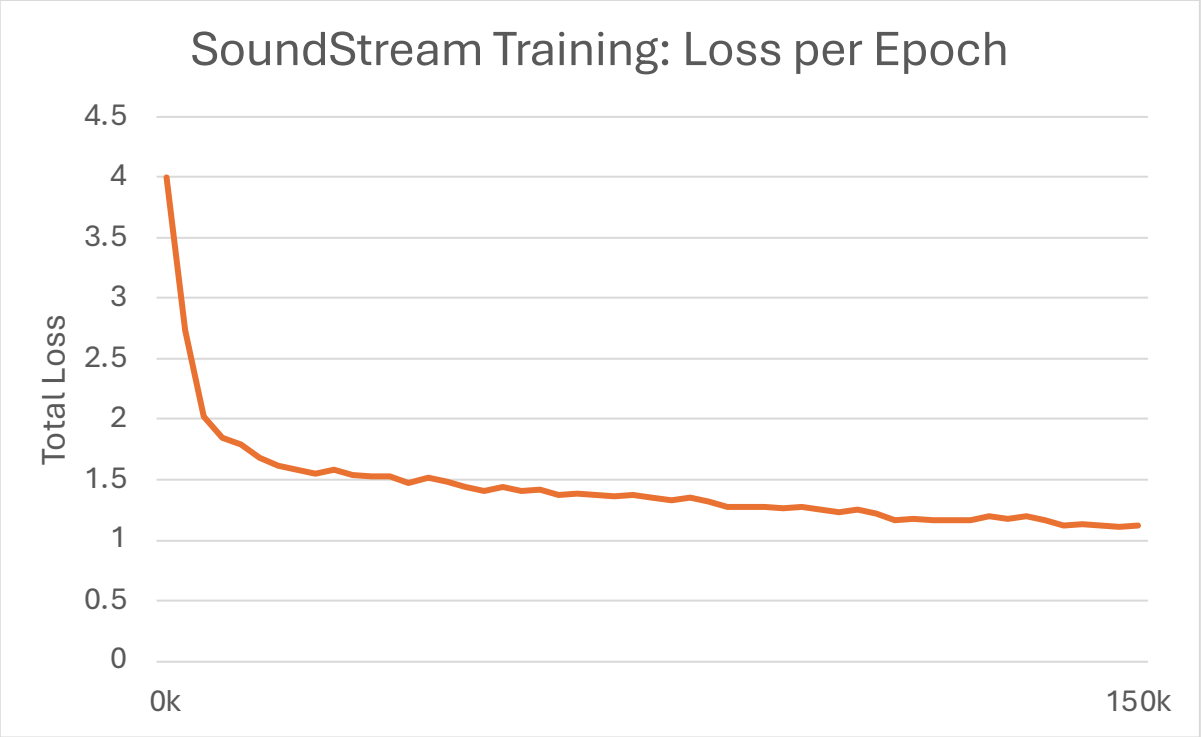


Figure 14. Graph for soundstream learning and loss rate across 150k epoch.

Figure 15 shows how the SoundStream model performed. Although there are some artifacts and noise, the original song could be distinctively heard. This was determined as sufficient before proceeding to the next step in training.

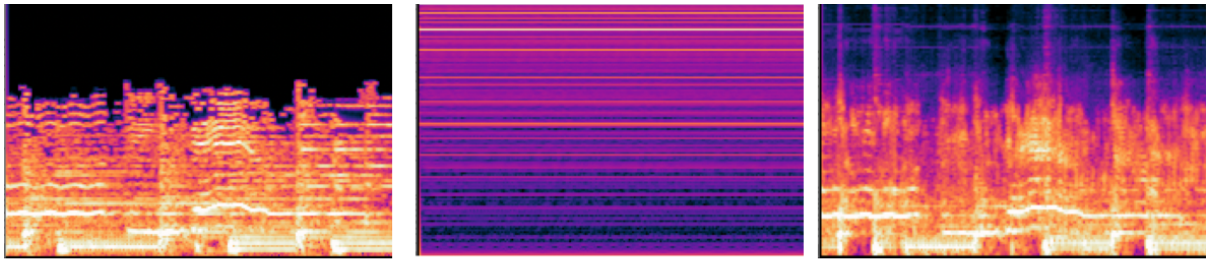


Figure 15. Mel-Spectogrram comparing the progress of SoundStream's succesful training.

5.2.1.2 MuLan and AudioLM

For the MuLan model, the recommendations on the GitHub are followed. For training, the Mulan asks for <audio, text> tensor pairing. The corresponding captions were padded to a desired length and turned to tensors as shown in figure 16. These where then stored to file.

```
for index in range(len(captions)):
    caption = (captions[index] * ((desired_length + len(captions[index]) - 1) // len(captions[index])))[:desired_length]
    audio, _ = librosa.load(os.path.join(audio_folder, f"{audio_ids[index]}.wav"), sr=1024, offset=0, duration=10)
    wavs.append(torch.tensor(audio).unsqueeze(0))
    texts.append(torch.tensor([ord(char) for char in caption]).unsqueeze(0))

wavs_tensor, texts_tensor = torch.cat(wavs, dim=0), torch.cat(texts, dim=0)
```

```
Set num_epochs to 100,000

For each epoch in the range of num_epochs:
    Initialize total_loss to 0
    For each index and corresponding wav_tensor_batch, text_tensor_batch in the zipped lists of list_wavs and list_texts:
        Compute the loss using the model mulan with wav_tensor_batch and text_tensor_batch on GPU
        Backpropagate the loss
        Update total_loss by adding the current loss value
```

Figure 16. Code snippets of tensor creators pairs for training MuLan and batch processing and training.

Unfortunately, after 100k words the model did not perform properly. A significant disparity between the requisite training data, estimated at 80 billion pairings, and the actual dataset size provided by MusicCaps, consisting of only 5,000 entries. Therefore the, under the time constraints, MuLan was not reproducible.

The idea evolved to attempting to use the original AudioLM's training and disregard MusicLM. The theory lied on simply removing the audio conditioning that MuLan provided as well as the quantizer and attempting to still train the transformers. Unfortunately, similar obstacles where faced. Despite a successful training of the audio codec, neither semantic, coarse, or fine transformers performed well enough to produce anything that sounded melodic. Figure 17 shows how, there is some evidence of trained and tuned model, but when listening to the spectrogram it sounds muddled and not distinct enough from prompt to prompt.

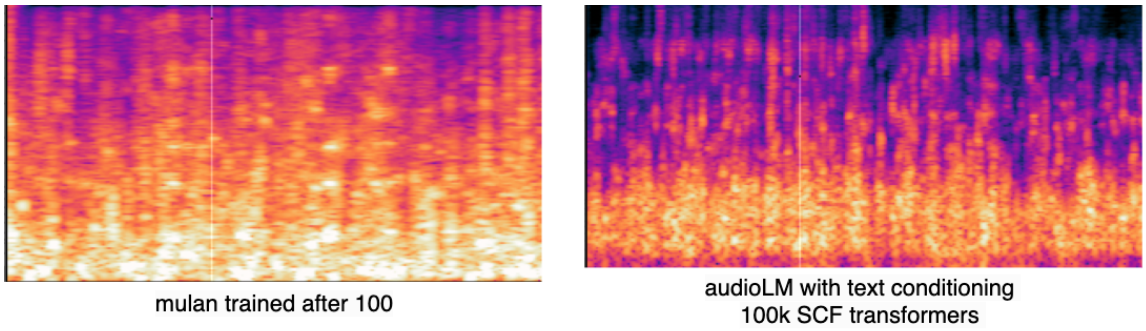


Figure 17. Mel-Spectrogram of failed MuLan training.

5.2.2 AudioLDM

The working implementation for the music generation uses AudioLDM, since this model is pretrained and works in full, there is no implementation for training. AudioLDM is available using Hugging Face Diffusers, facilitating access to pre-trained weights. The documentation offers different models to use, after comparing the different versions and running them, "audioldm-m-full" was used. This model stood out due to incorporating audio conditioning, which enhances the model's ability to generate music with higher fidelity and quality.

Checkpoint	Training Steps	Audio conditioning	CLAP audio dim	UNet dim	Params
audioldm-s-full	1.5M	No	768	128	421M
audioldm-s-full-v2	> 1.5M	No	768	128	421M
audioldm-m-full	1.5M	Yes	1024	192	652M
audioldm-l-full	1.5M	No	768	256	975M

Figure 18. Table provided by https://huggingface.co/cvssp/audioldm

Within the AudioLDM architecture, audio conditioning refers to shaping the audio output generated by the model, meaning incorporating additional information and features into the model to influence the generated audio's characteristics. Therefore "audioldm-m-full" can produce better quality features. Additionally, as provided in figure 18, the selected model distinguishes itself by having the highest CLAP audio dimensionality. This signifies the complexity and richness of the latent representation learned by the model during the pretrained phase. It can, therefore, capture a more intricate linguistic and semantic features. This is essential since our captioning will produce very abstract descriptions. Not very much related to musical qualities, i.e. it won't specify pitch, tempo, timbre… etc.

The model is integrated into the system by dividing the task into two major steps. The retrieval of the pipe from the diffuser model. It then tries to connect to an accelerator. Once the pipe is created, the audio is created as a NumPy array.

The parameters displayed in figure 19 were chosen due to the following:

- **num_inference_steps:** This determines the iterations during generation More inference steps typically result in higher-quality output but may also increase computational time. By settling the number on 20 implementation strikes a balance.
- **Guidance_scale:** adjusts the influence of the prompt on the text. Through thorough experimentation 3.5 was seen as fitting enough of the prompt characteristics without sacrificing quality.
- **Time_sec:** This will be adjustable when we mention the UI App implementation.

```
audio = pipe(prompt=prompt,
             audio_length_in_s=time_sec,
             num_inference_steps=20,
             guidance_scale=3.5,
             negative_prompt=negative_prompt,
             ).audios[0]
return audio
```

Figure 19. Code snippet of pipe creation parameters.

5.3 Application(UI)

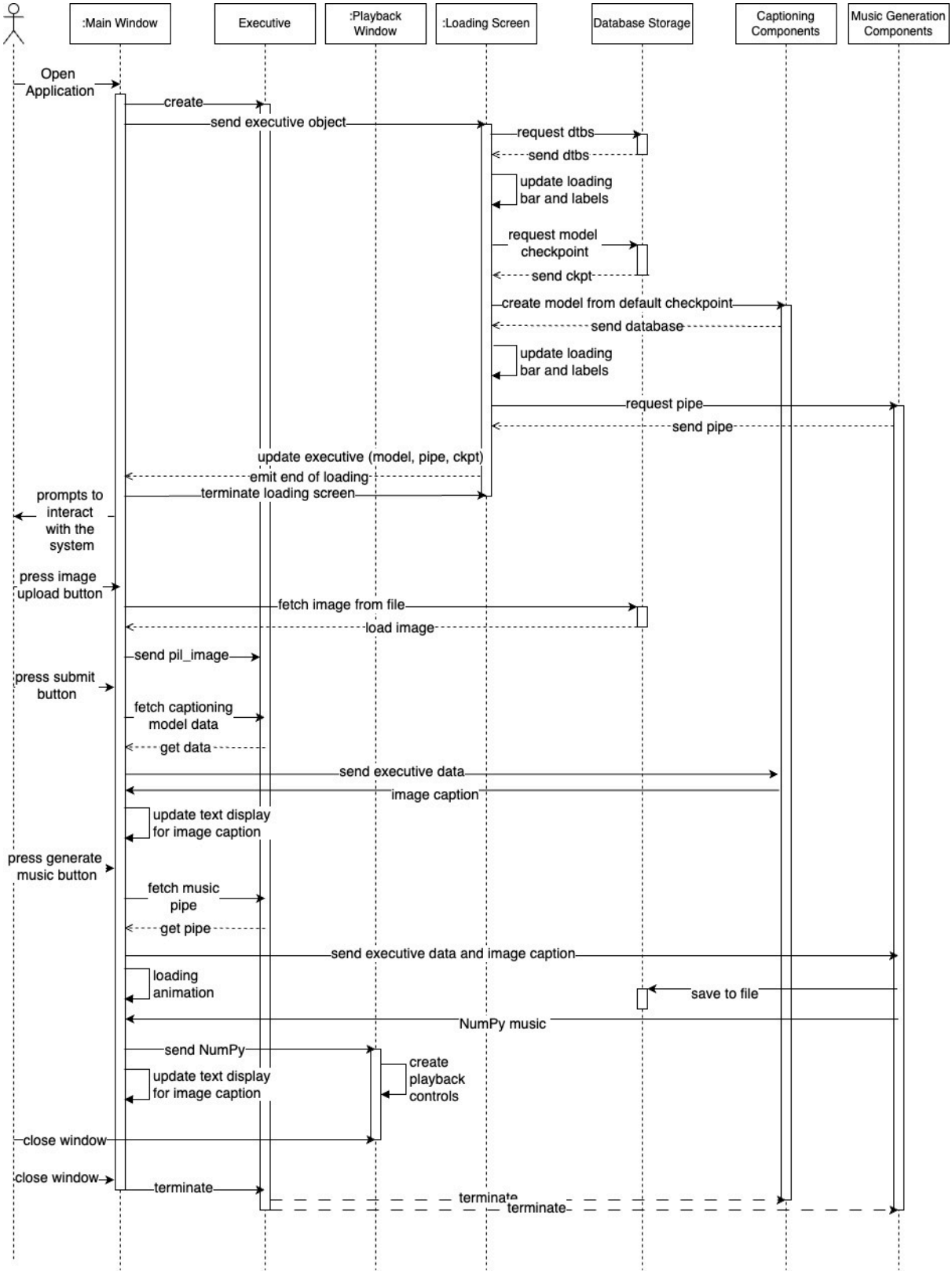


Figure 20. Sequence diagram of the default use of the system.

The UI and integration of the app followed closely the model of figure 20. Displayed below, figure 21 is a sequence diagram for the default usage of the system. The Captioning components as labelled encompass the CNN2RNN model and the prediction classes. Similarly, the Music Generation Components encapsulate the Diffuser models.

As discussed in the beginning sections, most of the processing is done in the loading stage, this is to, as mentioned, ensure that the user's interaction with the system is quick. Except for the music generation. Feedback is given in the form on an animation to help indicate that the system has not hung and is in fact working on the background.

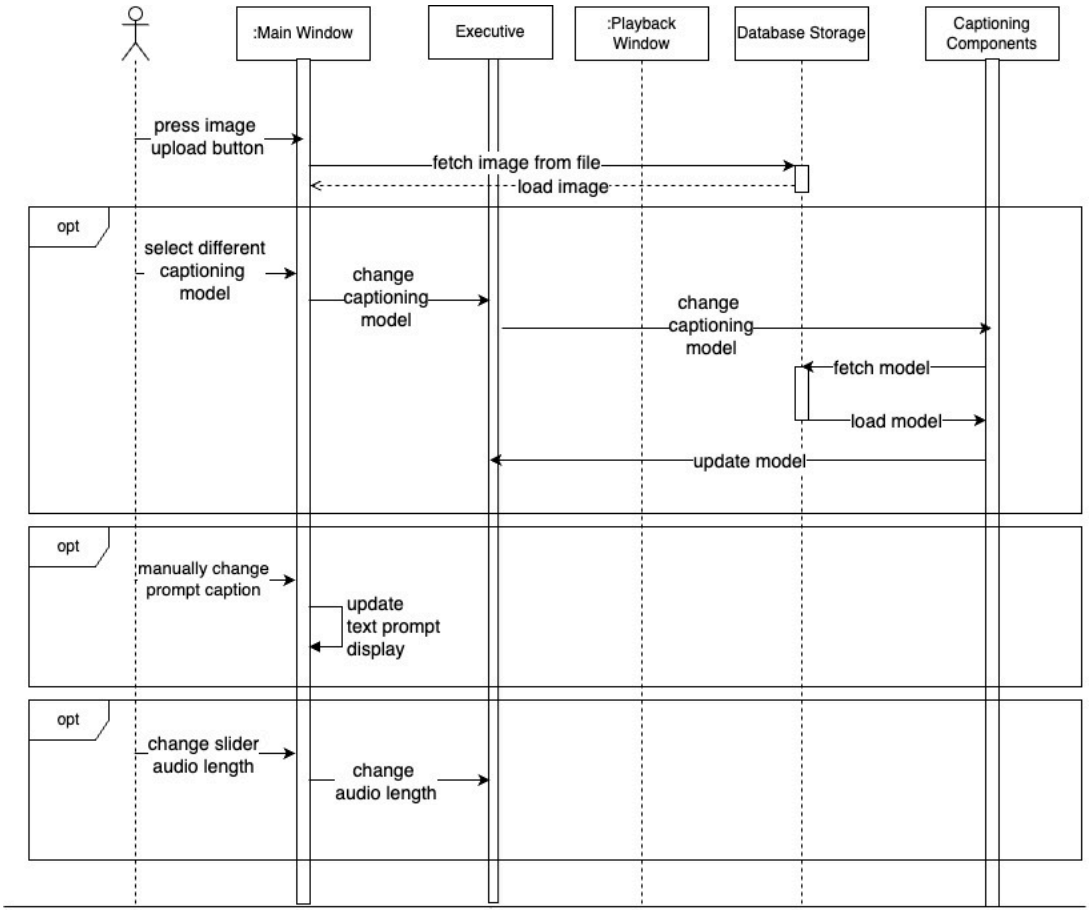


Figure 21. Sequence Diagram of optional/user customisation.

Figure 21 shows a sequence diagram outlining the customisable elements of the system. Key aspects are the upload of images, the changing of captioning models, the manual changing of the prompt generated with the user's own description and the length of music generated. This diagram served to summarise the understanding of implementing these features within the UI.

Displayed below, figure 23 shows an overview of the UI with figure 22 showcasing the visual interaction change from screen to screen.

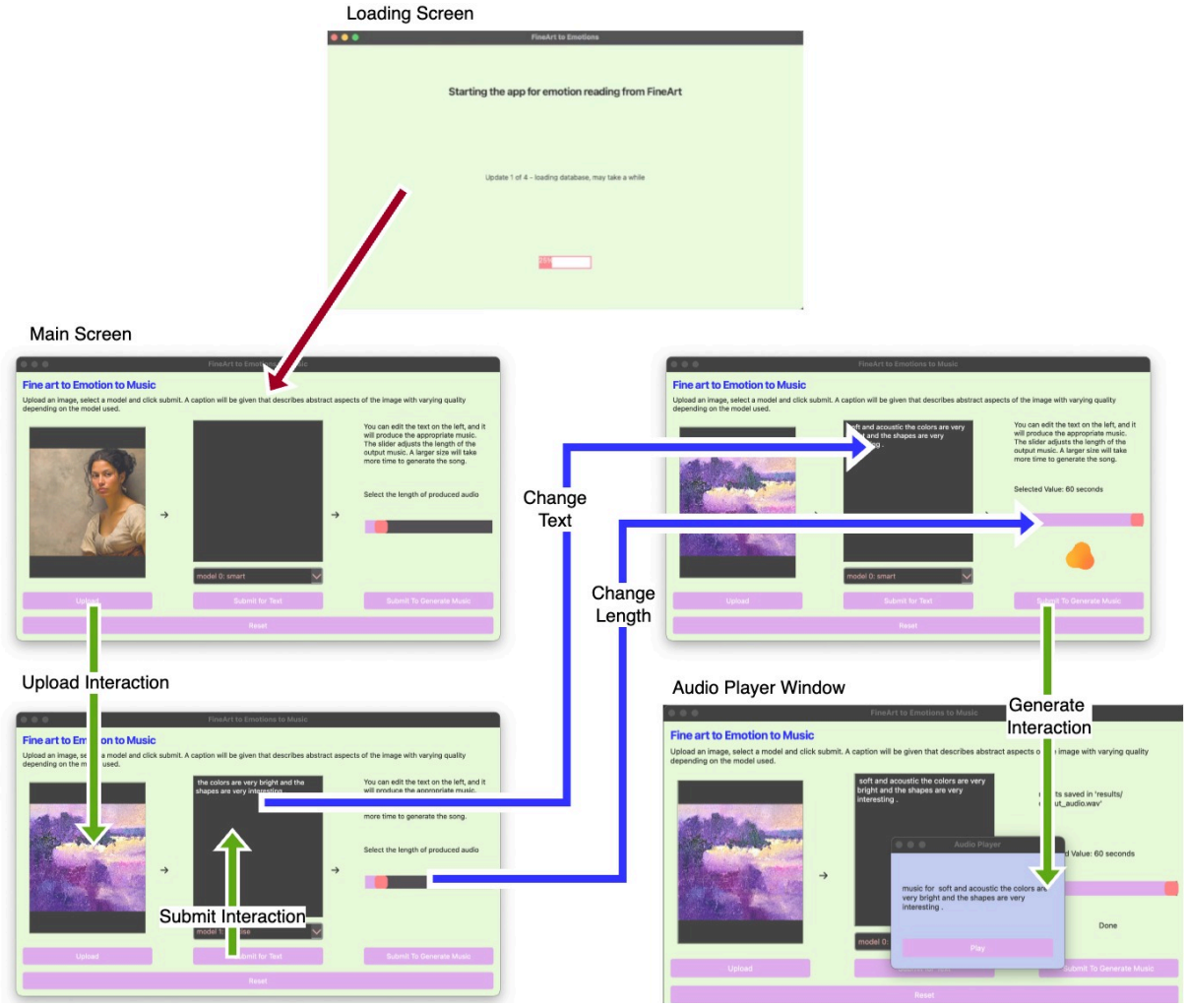


Figure 23. Flow of window and interactivity between screens and UI buttons and sliders.

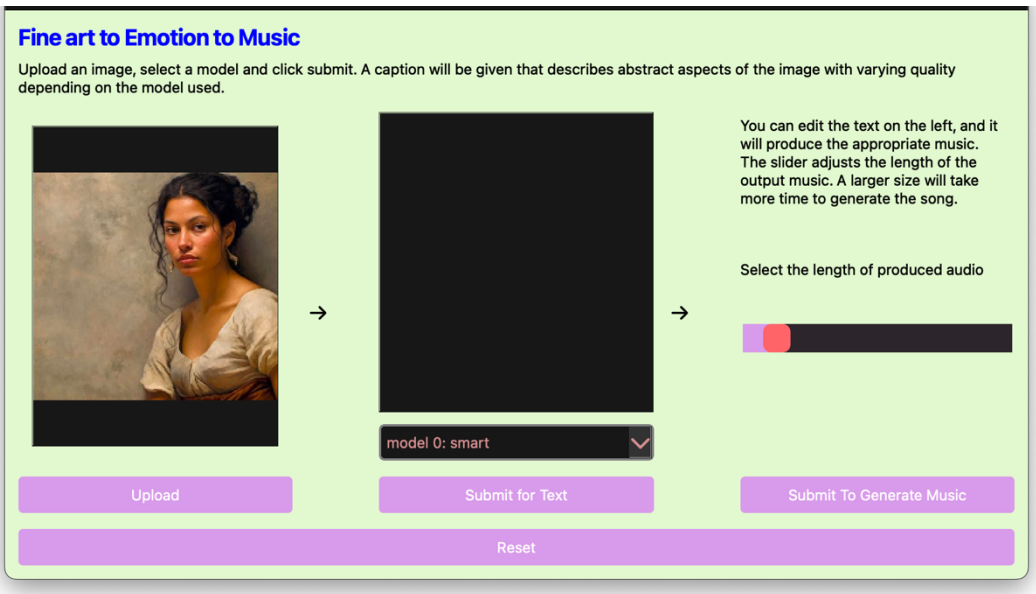


Figure 23. Display of the main window.

6 Testing and Evaluation

Since the nature of the results is subjective, evaluating the performance would require evaluation through by sampling a large extent of users. The constraints of the project did not allow for this, instead, for the image captioning model OpenAI was used to evaluate it. For the music generation, evaluation was less importantly. Since it uses pre-trained models by AudioLDM, instead the focus was placed on testing and assessing requirements met.

An overview of the training requirements was gathered by assessing how much it completed the Model Training Requirements laid out in Section 3. For Image Recognition and Emotion Captioning, both requirements where met. With a functioning system (accuracy evaluated below) and another model with altered parameters showing different results. For the Music Generation Integration, both requirements are filled. The system includes the option for accelerations in an MSP GPU. Moreover, the system tested different configurations to evaluate the best result. The ability to modify textual input is also available.

6.1 Canvas to Emotion: Model A

To start with, a new dataset of painting needed to be made. Rather than search for a database online, a new one was produced from google images. The idea behind this is to reproduce what potential users' expectations and how they may use the application. This approach aimed to mimic the real-world scenarios users might encounter when using the application. Therefore, the GoogleImageSearch library was used interfacing with Google's API, allowing for efficient retrieval of images based on specific keywords.

```
gis = GoogleImagesSearch(api_key, engine_id)
keywords = ['fine art','landscape', 'portrait', 'abstract', 'sad',
            'happy', 'contemporary', 'classic', 'nature', 'corporate']
for keyword in keywords:
  _search_params = {'q': f'{keyword} painting', 'num': 120}
  collection = []
  gis.search(search_params=_search_params)
  for image in gis.results():
      collection.append(image)
```

Figure 24. Code snipped of Google Collab's web-scrounge for new paintings.

Of the potential 1200 images searched, around 800 where suitable for the prospects of the project. A suitable image was deemed if it met the following criteria.

- High Quality
- Canvas based art (i.e. no pictures of sculptures, no pictures)
- The painting was at the forefront of the image.
- The painting was the only subject of the image.

Once the image database was produced. The CNN2RNN model was asked to produce a caption for each, these was stored in a csv. Subsequently, an API key was created to access OpenAI's API, specifically 'gpt-4-vision-preview' model. A request to OpenAI was passed with a header specifying the API key and a JSON string with the query and parameters.

```
response = requests.post("https://api.openai.com/v1/chat/completions",
                         headers=headers, json=payload)
answer = response.json()
```

Figure 25. Code snippet request for OpenAI

The JSON consisted of the model specificity text with the prompt and image (converted to base64. To limit the response output into a score, the token output was set to 5. After multiple tries with different prompt, this one produced a reliable scoring system.

```
“"would you say the painting EMOTIONALLY fits this description
"{desc}”? Score the Answer from 1 to 5. 5 being accurate, 1 being
inaccurate. Limit your answer to the number score."
```

Figure 26. Question asked to GPT-4 string for scoring.

Each score was stored parallel to the caption in the csv file, after 800 the results (figure 27). Scores of 1 were classified as inaccurate, 5 as accurate, and the in between as somewhat. The process was repeated for the second checkpoint for the CNN2RNN model.

6.1.1.1 Results

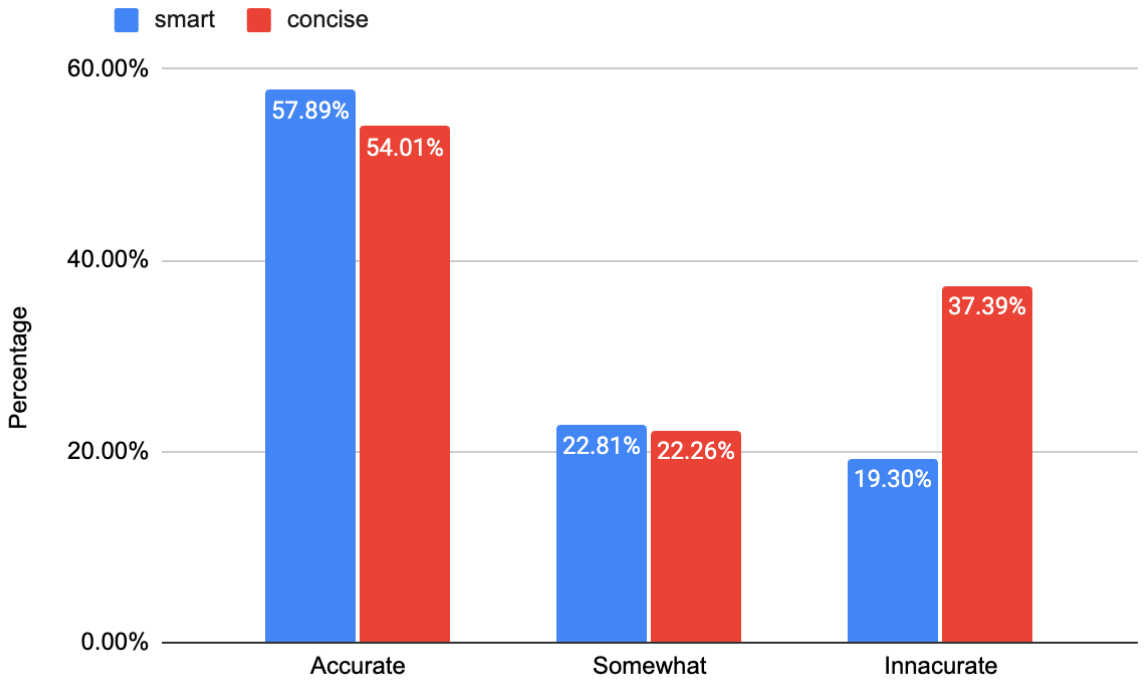


Figure 27. Chart showcasing GPT-4's evaluation of the captions.

The second trained model denoted as “smart” revealed a 58% exact match based on GPT4’s scoring system. Notably, 19% of the prompts were seen as entirely inaccurate. The second model trained (“smart”) produced similarly albeit with a higher expectation of failure in fully

matching prompts. Upon further examination, it could be determined that GPT-4 would score inaccurately, an example is with abstract or surrealist art. The nature of it is more up to varied interpretation and different conjecture tended to result in failures more frequently than expected (as depicted in Figure 28) where often it would harshly determine a painting as a failure.

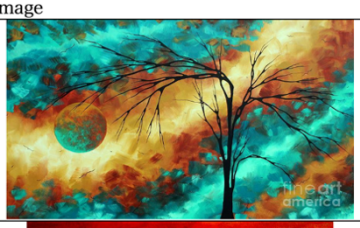
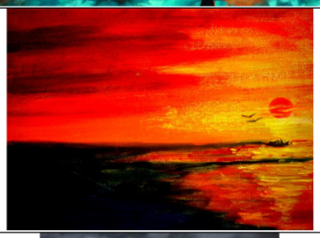
Image	Model A's Caption	Reason for OpenAi's inaccuracy
	the dark colors looking down makes me feel sad .	This painting has elements that may feel sombre, such as the dying tree
	the dark clouds and the dark clouds make me feel like I 'm in a boat .	There is a boat in the image, although small. Colours are generally dark (there is a lot of negative space).
	the dark colors and the way the people are looking at the viewer makes me feel like i 'm in a horror movie .	The elements in the painting are surreal, the environment is a bit tetric. Thus, invoking feeling of a horror movie is justified

Figure 28. Table showing selected examples of the failure of OpenAI to correctly score.

Regardless, this type of discrepancy did not happen enough to significantly skew the overall accuracy of the captions. To understand better the results, a random sample of 120 labelled as entirely inaccurate was selected and analysed. Of all inaccurate scores, 48% contained the word 'man', of those most had words with negative connotations such as 'dark' and 'kill'.

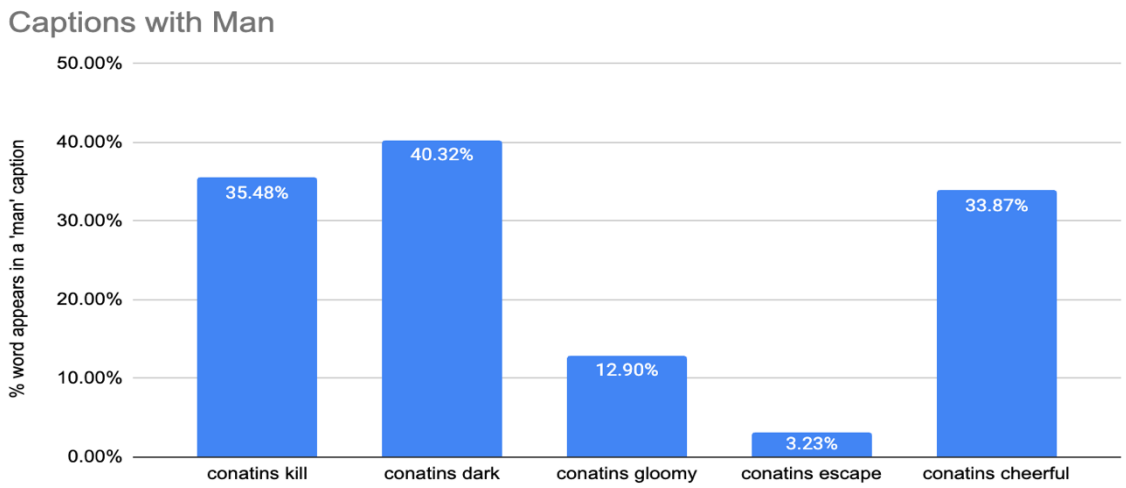


Figure 29. Analysis of keywords within captions containing 'man'

Looking at the images individually, it appears that the model has defaulted to taking unseen data or confusing art as a representation of ‘man’ followed by closely predicted subsequent words. Examples of this phenomenon are illustrated in Figure 30. One plausible explanation could be that the dataset provided by ArtEmis relies heavily on classical painting of the Western world, leading to misclassifications of subjects such as black women due to a "default" pitfall in the model's training data.

Image 1	Image 2	Model A’s Caption	Evaluation
		the man looks like he is trying to kill the man in the middle of the painting .	A variation of this caption was repeated 10 times from the 86 captions containing ‘man’ The first image is completely incoherent with the caption. The second is an example where a man would be identifies in the painting, but it would label it as aggressive.
		the man 's eyes are very expressive, and the colours are very dark and gloomy .	Wrongly identifies as dark and gloomy
		the man looks like he is trying to escape the man	The image depicts horses running not men trying to escape.

Figure 30. Sample of the CNN2RNN failure at capturing correctly.

Another possible explanation for the default being men centred around negative emotions could be the inherit biases in the dataset. If the dataset predominately portraited men in aggressive rolls, such as war paintings, then when the model cannot capture a clear contradiction, it will classify it as darkness, anger, or a form of attack. Furthermore, the observed behaviours may also be attributed to a convergence to a local minimum, where the model settles into a suboptimal solution that fails to generalize unseen data. In that scenario, the association to “man” would have been underscored by specific features, although inaccurate.

6.2 Application (UI)

The Application's UI was evaluated based on how much it completed the Functional and Non-Functional requirements. It was tested with friends and family, and they were each asked if they found the requirements to be fulfilled. For Functional Requirements points where completed. Non-Functional Requirement 1 was a contest for debate. The 'efficiency' of the music generation should not be agreed upon whether it was quick or not, as some considered speed while others focused on other aspects.

Moreover, the scalability of the system was another point of contention. Theoretically, more captioning models could be added, but the music generation would require a change in architecture. Mainly the fact that the pipe is loaded at the beginning of the App's run. Adding a change in music generation models would require a system to reload a different diffuser (for example). This adding an additional waiting time. Nevertheless, the visual feedback of 'loading' is currently threaded, therefore, although possible to scale music models, it would be costly. The primary struggles with meeting the non-functional requirements come with compatibility. The system was developed using Mac OS, and cross-compatibility with other operating systems could not be thoroughly tested within the project timeline. Despite the limitation, the code does include robust error handling mechanisms, such as tr-catch blocks with guides as to how to make the system properly. Overall, all other system requirements where met. Most importantly, all 'must' requirement where fulfilled. Figure 31 illustrates an example of UI interactive testing.

	Test Case ID	Description	Expected Result	Pass/ Fail
Image Upload	TC-01	User uploads an image file	Image is successfully uploaded and displayed	**Pass**
	TC-03	User tries to upload multiple images	System prompts to upload only one image	**Pass**
	TC-04	User uploads a large image file	System maintains ratio but downsizes it within the display box	**Pass**
Image Captioning	TC-05	User selects a captioning model	Model selection dropdown displays options	**Pass**
	TC-06	User inputs text for captioning	Text is displayed in the input field	**Pass**
	TC-07	User generates captions for uploaded image	Captions are generated and displayed	**Pass**
Music Generation	TC-08	User initiates music generation	System starts generating music	**Pass**
	TC-09	User controls music playback (play, pause, stop)	Playback controls function as expected	**Pass**
	TC-10	User adjusts music generation duration	Slider changes music duration accordingly	**Pass**
Interactivity/Feedback	TC-11	Visual feedback provided for music generation	Display loading icon	**Pass**
	TC-12	Loading screen displayed during processing	Loading screen appears during heavy tasks	**Pass**
	TC-13	Reset button clears text and images	Text and images are reset to default state	**Pass**
Integration Testing	TC-14	Generated captions passed to music generation	Captions are correctly inputted for music	**Pass**
	TC-15	UI displays the correct cation used in the music generation with a keyword at the beginning to create music	The Display shows "music for " and then the captioning/text input	**Pass**
	TC-16	Captioning and music generation interact smoothly	Captioning output influences music generation	**Pass**

Figure 31. Test Table for UI features.

7 Project Management

At the initial conception of the project a Gantt chart was produced. Figure 32 is a simplified version of it, adjusted to the actual outcome of the timeline, the full version will be available in the Git Repository.

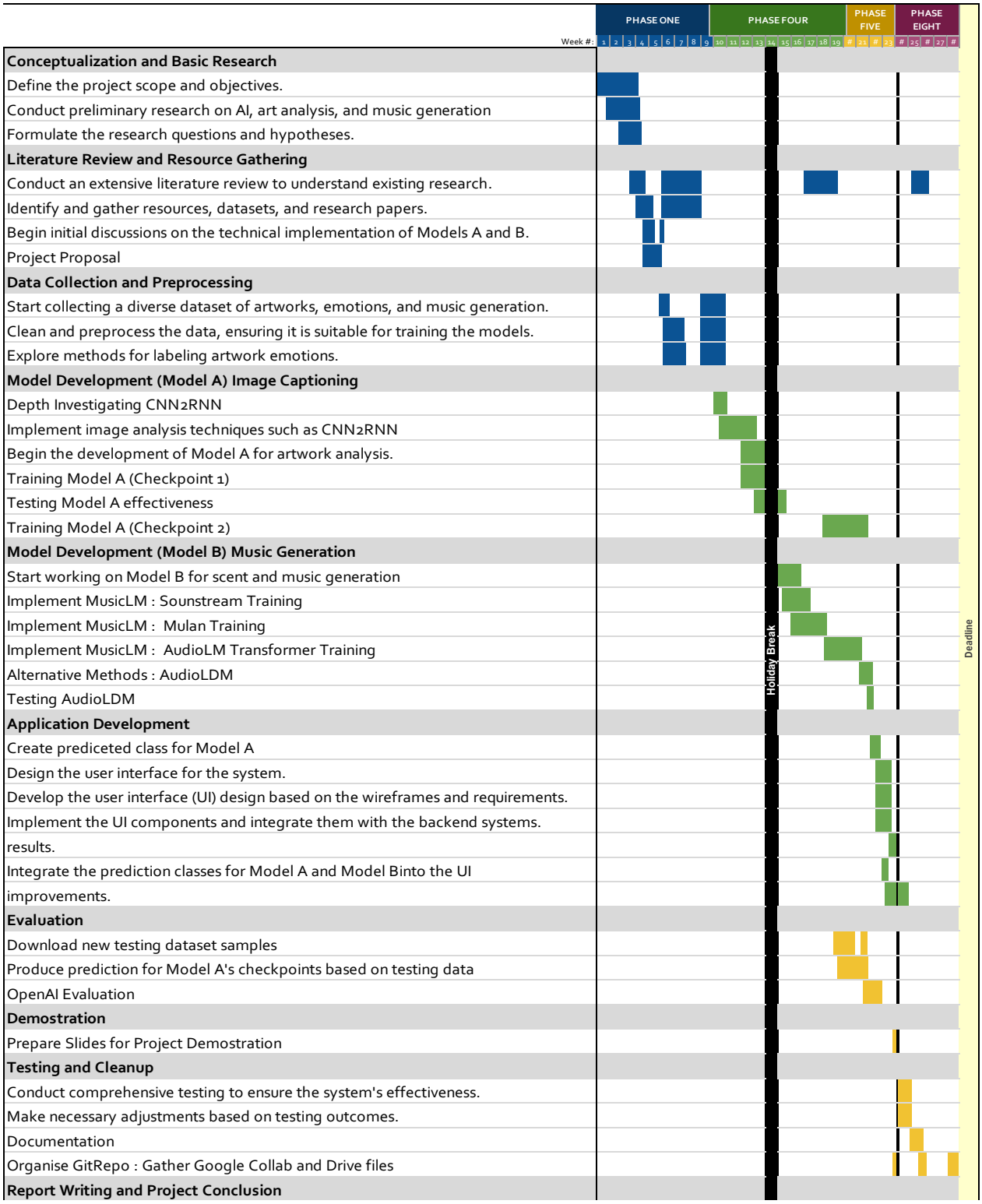


Figure 32. Gantt chart summary version.

7.1.1 Milestones

The timeline was slit into these key milestones:

I. Conceptualization and Basic Research
II. Literature Review and Resource Gathering
III. Data Collection and Pre-processing
IV. Model Development (Model A) Image Captioning
V. Model Development (Model B) Music Generation
VI. Application Development
VII. Evaluation
VIII. Demonstration
IX. Testing and Clean-up
X. Documentation
XI. Report Writing and Project Conclusion

7.1.2 Setbacks

An initial setback came with milestone III. The Data provided by ArtEmis, although extensive, was not consistent with Wiki-Art's most recent layout and painting identification. Issues spotted included, from the string identifying a piece of art in the CSV file:

<<artist_name>>_<<painting_name>>_<<year>>_<<extra>>

- The <<extra>> field at the end would not correctly correlate to any identifiable keywork for the art. Sometimes it would be a random number, probably as a listing for painting of the same name and year.
- Artist names or painting with any accents or diacritics would have been encoded wrongly, and exact decoding could not be consistently found, with some needing decoding using utf-8, other times utf-16. However, this still did not solve the issue with most.
- A wrongly cited date at the of the file being made but corrected in present time in
- Moreover, the availability of painting depended too on whether they were available to the country.
- Originally the coding was done with validation in mind, where, if one criterion for painting search showed to viable results, another would be tried. The issue with this is that often WikiArt would list a URL as available with a SRC image in the same class as the targeted portrait, but the painting itself would be a blank default image.

When dealing with so many entries, issues like this often escaped through the initial testing. Moreover, halfway through the downloading process, WikiArt updated their page layout. Meaning that the original SRC image fetch no longer corresponded to the paining targeted. All these issues were solved by expanding the search outside of WikiArt. Google Search was used (as seen in the design section) prioritising URL request with WikiArt in them. Moreover, for the image extraction, the image tags were compared rather than looking for a specific order. This yielded in a substantial, accurate image sample.

Despite planning, significant setback were encountered during milestone V, when it came to the failed reproducibility of MusicLM and AudioLM. These challenges necessitated adaptability and a revaluation of the project priorities. Efforts were directed into extensively researching music generation through semantics. The idea was to, even if the software and

ML wouldn’t work, that an in-depth understanding could be demonstrated. Fortunately, within the research AudioLDM was found and implemented into the system.

This approach was enabled by the agile project management practices and regular process assessment. The iterative approaches of agile allows for flexibility and parallel work within the system. Meaning, despite the potential of a failed music generation model, model A could still be reinforced. Moreover, the emphasis on producing software on the early stages aligned well with solo development, allowing for swift assessment of progress and adjustment of expectations and goals along the way. Therefore, despite ethe major setback, the project was able to quickly adapt.

8 Conclusion

This paper presents a comprehensive exploration of an integrated system for image captioning to music generation, aimed at , emotional language and context to artwork. Through the development and testing phases key insights where gained, shaping the project's impact and suggested future direction.

8.1.1 Impact

The development of the system reflects an understanding of the current state in the fields of AI-driven creative generators, offering users a novel tool to translate one AI assessed art medium to another. Thus, allowing for new avenues of artistic exploration and understanding of what ML methodologies are currently providing this access. Although the model showed bias in captioning, integral aspects of emotion analysis was still deemed sufficiently accurate.

8.1.2 Future Steps

Firstly, to improve the accuracy of results, obtaining a diversity of image captions could evolve refining the training data and model architecture better. Adopting pre-existing models such as Style-Net to enhance the image output would also be implemented. Or, training a system to identify key emotions of the painting and the using NLP to generate a caption reliant on that emotion, therefore preventing the 'default' pitfall explored in the evaluation section. Additionally, exploring alternative music generation approaches, such as generative adversarial network, could enhance the system's ability to vary musical qualities such as rhythm, timbre and pitch.

8.1.3 Final Statement

The integrated system represents pioneering effort in leveraging AI to infuse emotional depth into creative outputs. By combining an understanding if the current state of the art and unifying these aspects, the system offers a unique platform for expression an emotional storytelling. While there are challenges and opportunities on the horizon for AI, the contribution of ML as a catalyst for expression amongst human and a tool for critical thought is also undeniable.

9 References

[1] Aladdin Persson, “PyTorch Tutorials,” YouTube. Jun 2021.

[2] A. M. Belfi, B. Karlan, and D. Tranel, “Music evokes vivid autobiographical memories,” *Memory*, vol. 24, no. 7, pp. 979–989, Aug. 2016, doi: 10.1080/09658211.2015.1061012.

[3] L. L. Cuddy, R. Sikka, K. Silveira, S. Bai, and A. Vanstone, “Music-evoked autobiographical memories (MEAMs) in alzheimer disease: Evidence for a positivity effect,” *Cogent Psychol*, vol. 4, no. 1, 2017, doi: 10.1080/23311908.2016.1277578.

[4] E. Rivas Ruzafa, “Pix2Pitch: generating music from paintings by using conditionals GANs,” *ETSI_Informatica*, 2020.

[5] M. J. Wilber, C. Fang, H. Jin, A. Hertzmann, J. Collomosse, and S. Belongie, “BAM! The Behance Artistic Media Dataset for Recognition Beyond Photography,” Apr. 2017, [Online]. Available: http://arxiv.org/abs/1704.08614

[6] C. Gan, Z. Gan, X. He, J. Gao, and L. Deng, “StyleNet: Generating attractive visual captions with styles,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Institute of Electrical and Electronics Engineers Inc., Nov. 2017, pp. 955–964. doi: 10.1109/CVPR.2017.108.

[7] P. Kashyap, S. Phatale, and I. Drori, “Prose for a Painting,” Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.03634

[8] N. Garcia and G. Vogiatzis, “How to Read Paintings: Semantic Art Understanding with Multi-Modal Retrieval,” Oct. 2018, [Online]. Available: http://arxiv.org/abs/1810.09617

[9] P. Achlioptas, M. Ovsjanikov, K. Haydarov, M. Elhoseiny, and L. Guibas, “ArtEmis: Affective Language for Visual Art.” [Online]. Available: https://artemisdataset.org.

[10] A. Agostinelli *et al.*, “MusicLM: Generating Music From Text,” Jan. 2023, [Online]. Available: http://arxiv.org/abs/2301.11325

[11] Z. Borsos *et al.*, “AudioLM: a Language Modeling Approach to Audio Generation,” Sep. 2022, [Online]. Available: http://arxiv.org/abs/2209.03143

[12] Y.-A. Chung *et al.*, “W2v-BERT: Combining Contrastive Learning and Masked Language Modeling for Self-Supervised Speech Pre-Training,” Aug. 2021, [Online]. Available: http://arxiv.org/abs/2108.06209

[13] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, “SoundStream: An End-to-End Neural Audio Codec,” Jul. 2021, [Online]. Available: http://arxiv.org/abs/2107.03312

[14] H. Liu *et al.*, “AudioLDM: Text-to-Audio Generation with Latent Diffusion Models,” Jan. 2023, [Online]. Available: http://arxiv.org/abs/2301.12503

[15] B. Elizalde, S. Deshmukh, M. Al Ismail, and H. Wang, “CLAP: Learning Audio Concepts From Natural Language Supervision,” Jun. 2022, [Online]. Available: http://arxiv.org/abs/2206.04769

[16] “Advanced Guide to Inception v3.” [Online]. Available: https://github.com/tensor:ow/tpu/tree/master/models/experimental/inception

[17] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, “HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units,” Jun. 2021, [Online]. Available: http://arxiv.org/abs/2106.07447

10 Appendix

Instructions for Repository

Clone the Git repository : https://git.cs.bham.ac.uk/projects-2023-24/mar933

Follow the instruction on figure 33 to setup the system.

Downloading NLK via pip

1. **Install NLTK via pip:**

```
pip install nltk
```

Setting up Venv

1. Create Virtual Environment:

   Open a terminal and navigate to your project directory. Run the following command to create a virtual environment named 'venv':

```
python -m venv venv
```

2. Activate Virtual Environment:

   On Windows:

```
.\venv\Scripts\activate
```

   On Unix or MacOS:

```
source venv/bin/activate
```

3. Installation of Dependencies:

   With the virtual environment active, install project dependencies using:

```
pip install -r requirements.txt
```