

A Multimodal Large Language Model for Music Captioning



AALBORG UNIVERSITY STUDENT REPORT

MASTER PROJECT
MATHEMATICAL ENGINEERING

Authors:

Anders Lauridsen
Jacob Mørk
Jakob Olsen

Supervisors:

Zheng-Hua Tan
Sven Ewan Shepstone
Pablo Martinez-Nuevo
Francesc Lluís Salvadó

June 3, 2024



Department of Mathematical Sciences

Skjernvej 4A

Aalborg University

<http://www.math.aau.dk>

AALBORG UNIVERSITY

STUDENT REPORT

Title:

A Multimodal Large Language Model for Music Captioning

Theme:

Natural Language Processing

Project Period:

spring-semester 2024

Project Group:

{ahll19, jmark18, jols19}@student.aau.dk

Authors:

Anders Lauridsen

Jacob Mørk

Jakob Olsen

Supervisors:

Zheng-Hua Tan

Sven Ewan Shepstone

Pablo Martinez-Nuevo

Francesc Lluís Salvadó

Number of Pages: 40

Date of Completion:

June 3, 2024

Abstract:

In this project, the goal was to implement a multimodal model, using an audio encoder and a large language model, capable of creating music descriptions given a song. A multimodal converter model was developed for captioning 10 second music clips. The model was consistently able to generate descriptions, however, it struggled with hallucinations and inaccuracies. The model's performance was measured using the BERTScore and a qualitative evaluation. Future work should prioritize fine-tuning the large language model together with the audio projection layer to combat the current issues. Hereafter, further research should look into other language models, improve the prompt used, and try different audio encoders.



Anders Lauridsen



Jacob Mørk



Jakob Olsen

Preface

This project was written in the period from February 1st, 2024 to June 3rd, 2024 by three Mathematical Engineering students at Aalborg University in the final semester of the Master's. We would like to thank supervisors Zheng-Hua Tan, Sven Ewan Shepstone, Pablo Martinez-Nuevo, and Francesc Lluís Salvadó for their support throughout the project period.

In this project, it is assumed that the reader has an understanding of the subjects presented during study for the Bachelor's and Master's degrees in Mathematical Engineering from Aalborg University.

Aalborg University, June 3, 2024

Contents

1	Introduction	1
1.1	Problem Statement	3
2	Natural Language Processing	4
2.1	Transformer Architecture	4
2.2	Mistral 7B	7
2.3	Prompting	7
2.4	Evaluation	9
2.4.1	BERTScore	9
3	LoRA and Quantization	12
3.1	LoRA	12
3.2	Quantization	13
3.2.1	QLoRA	14
4	Multimodal Models	16
4.1	Multi-Window Masked Autoencoder	17
4.2	J2A-2.x Implementation and Architecture	17
5	Experiment	21
5.1	Quantitative Evaluation	21
5.2	Qualitative Evaluation	23
6	Discussion	28
6.1	Further Research	29
7	Conclusion	31
	Bibliography	32
A	J2A-1.0	36
A.1	Features	36
B	Training Configurations	37
C	Prompt	38
D	J2A-2.x Responses	39

1 | Introduction

The music industry has undergone several years of consecutive growth. Much of this growth can be attributed to the advent of music streaming, which represented 65% of the music industry’s revenue for 2021 [1]. Recent advances have been made in music-related machine learning, such as the advent of the Suno¹ platform. However, music-related tasks for modern machine learning models have not been researched as much as tasks related to text and images [2].

Music-related machine learning models are of interest to companies that produce headphones, speakers, televisions, soundbars, and more. To improve user experience, these products often come with a method for the user to adjust the equalization of the products. Equalization tools might be too complicated for many users. Therefore, a machine learning model, which helps users adjust the equalization, might be of interest to such companies. A natural way to interact with such a model, without removing all agency from the users, is to utilize modern advancements in Natural Language Processing (NLP). An illustration of this type of model can be seen in Figure 1.1.



Figure 1.1: A simple model that takes a text prompt as input and returns equalization parameters.

One reason for the increased interest in NLP is the development of a new model architecture called the transformer [3]. This model architecture excels in handling long-range dependencies and is therefore very useful for language tasks. Language models that build on the transformer architecture, which are colloquially known as Large Language Models (LLMs), have driven the increased interest in multimodal models. A multimodal model is a model that can handle two or more modalities [4].

In [2] they present the LLark model, a multimodal model, which is able to solve three different music-related tasks. These tasks are defined as music understanding, music captioning, and music reasoning. Music understanding is the task of extracting a single and generally global property of a piece of music, such as the tempo or the key of a song. Music captioning is the task of summarizing a piece of music in a plain text format. Music reasoning is the task of combining multiple pieces of information about that piece of music, or relating information about the music to information external to that piece of music.

¹<https://suno.com/about>

To improve the system from Figure 1.1 we propose a multimodal model shown in Figure 1.2. This proposed change adds the music as context for the model, which may improve the process of determining the equalization parameters. The task of the sound descriptor of our proposed model is analogous to the music captioning task from the LLark model [2].

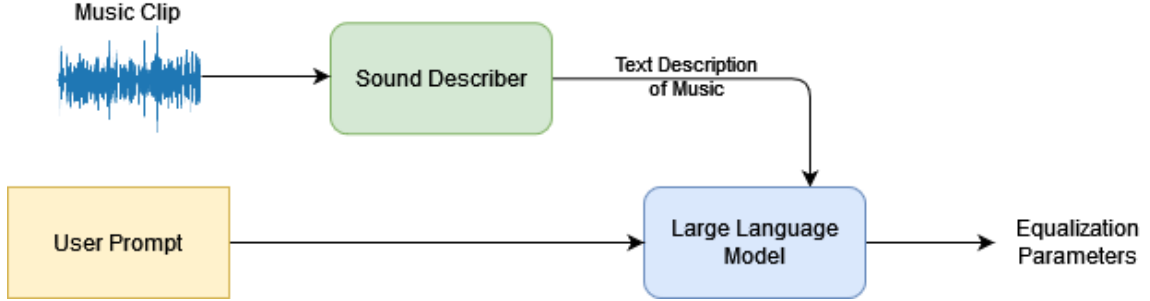


Figure 1.2: Two combined machine learning models. The sound descriptor is a multimodal model, which takes a sound clip as input and returns a text description of the given input. The second model is an Large Language Model (LLM) which takes a user prompt and a music text description as input, and returns fitting equalization parameters.

The sound descriptor in Figure 1.2 can be designed in many different ways. One way is to use a multimodal converter [5], which is the approach used in the LLark model. The structure of our proposed sound descriptor is illustrated in Figure 1.3.

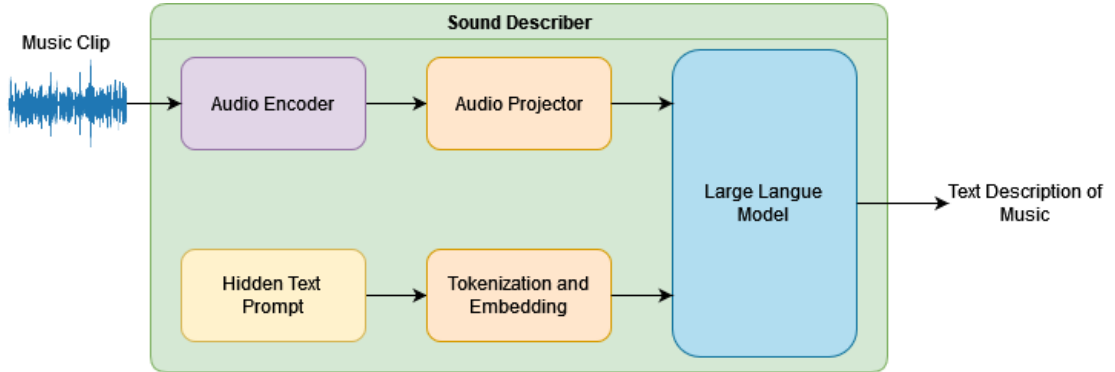


Figure 1.3: The figure shows a multimodal converter setup. The model handles the audio modality by connecting the audio encodings to the Large Language Model (LLM) using a projection layer. Furthermore, the text prompt is tokenized and embedded to be passed into the LLM. The output of the LLM is then a description of the music clip. The prompt instructs the LLM to solve the task of music captioning and is not accessible by the user.

One of the challenges in developing multimodal models for audio is the difficulty in obtaining large high-quality, richly annotated music datasets. The largest sets of such data we have been able to find, which are also used in [2], are YouTube8M-MusicTextClips [6] and MusicCaps [7]. YouTube8M-MusicTextClips provides over 4 thousand 10 second music clips captioned by the YouTube video annotation system and MusicCaps has over 5 thousand 10 second music clips captioned by human experts.

To develop a model capable of describing music clips, we test multiple approaches to the problem. This way we can compare performance between the models and set a direction for further development on the topic.

1.1 Problem Statement

This project aims to develop a multimodal model that can generate captions for short music clips. To achieve this, we propose the following research question:

Using an audio encoder and a large language model, is it possible to create a multimodal model that can generate captions for music clips of 10 seconds?

2 | Natural Language Processing

Natural Language Processing (NLP) has been a large area of research for many years. The field has made a large leap with the advent of the transformer architecture which was first proposed in [3]. The transformer was proposed in order to rectify the issues associated with Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) Networks, and Gated Recurrent Neural Networks (GRNNs). Some of these issues include vanishing gradients and the non-parallelizable process by which inference is carried out. These issues impede the size of the context that the model can be "aware" of, and the speed of inference and training of the model.

The Mistral 7B model is a transformer model which can be used for generating natural language [8]. This Large Language Model (LLM) uses the transformer structure as the one proposed in [3] but with a few modifications made to increase the inference speed. To interact with an LLM, prompting is used for having a text-based "conversation" with the model. To evaluate the quality of an LLM text output there exist many methods as this is not a straightforward task. Methods for evaluating an LLM include making humans evaluate the text output of the LLM or using text distance measures between captions (what is considered ground truth in this type of experiment) and the text output from the LLM.

In Section 2.1 we describe the general structure of the transformer models. In Section 2.2 we introduce the Mistral 7B model, and the modifications they have made to the traditional transformer architecture proposed in [3]. In Section 2.3 we introduce the concept of prompting and different strategies that can be used for interacting with the LLM. Lastly, different ways of evaluating an LLM are introduced in Section 2.4.

2.1 Transformer Architecture

Transformers are purely attention-based models, which can make use of an encoder-decoder structure. The encoder and decoder both consist of multiple layers, which consist of Feedforward Neural Networks (FNNs), Multi-Head Attention (MHA), and normalization sublayers, the decoder then has an additional sublayer, namely a masked MHA sublayer. An illustration of the architecture can be seen in Figure 2.1.

2.1. TRANSFORMER ARCHITECTURE

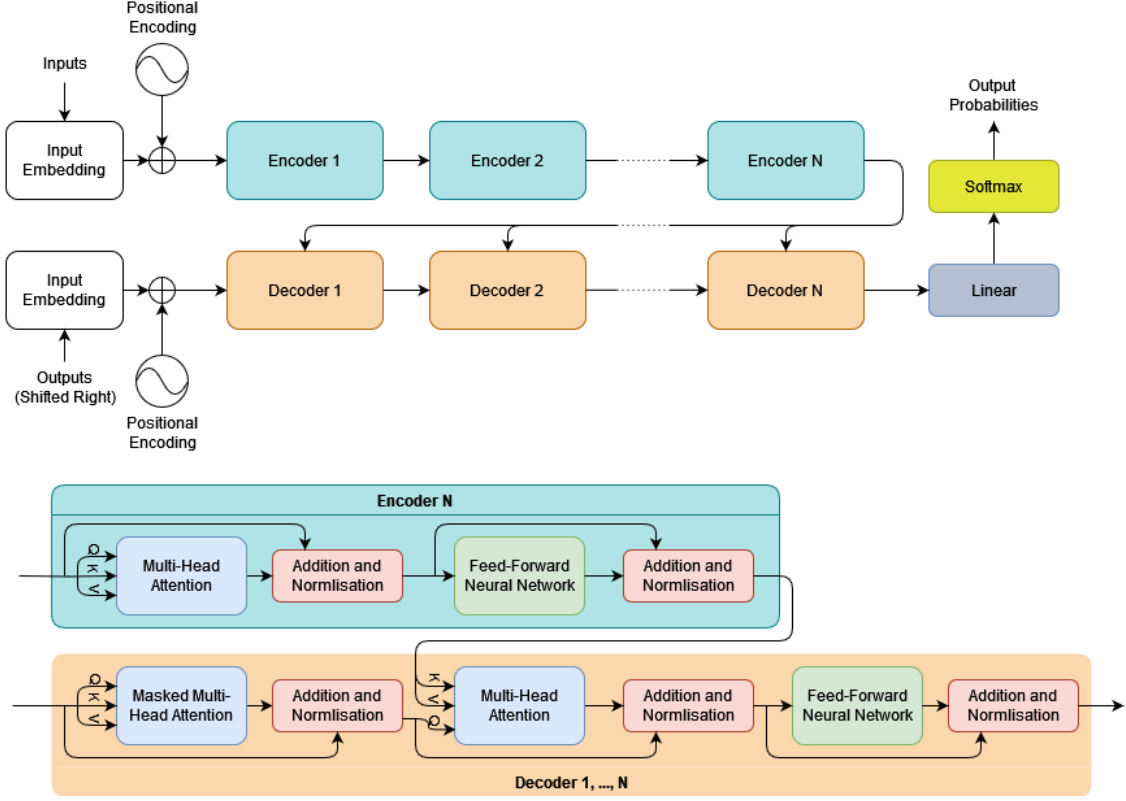


Figure 2.1: An illustration of a transformer using the encoder-decoder structure. Here the Q , K , and V are the query, key, and value matrices. The connections skipping the Multi-Head Attention (MHA) and Feedforward Neural Network (FNN) sublayers are residual connections. The output shifted right is the target sequence with a starting token in front.

The encoder-decoder structure is designed to allow for variable input and output lengths, as compared to a traditional Deep Neural Network (DNN) such as a large FNN. Using an FNN one has to define the input sequence length and the output sequence length. Using an encoder and decoder structure allows for arbitrary sequence lengths in the sequence-to-sequence type task which the transformer was designed to solve [3]. The variable sequence length is made possible in the encoding stage by updating a hidden internal state, which represents the input to the model. The variable output length is made possible by the decoder choosing tokens based on the hidden state, and only stopping once a special end-of-sequence token has been reached.

In Figure 2.1 it can be seen that the input is first embedded, where after a positional encoding is added. The positional encoding is needed since transformers do not have any positional knowledge, unlike RNNs, which have knowledge of other tokens in a sequence from the recurrence of the model.

There are multiple ways of implementing positional encoding, both learned and fixed. In [3] they implement the following positional encoding

$$PE(x, i) = \begin{cases} \sin\left(\frac{x}{10000^{2i/d_{model}}}\right), & \text{if } i \text{ even,} \\ \cos\left(\frac{x}{10000^{2i/d_{model}}}\right), & \text{if } i \text{ odd,} \end{cases} \quad (2.1)$$

where x is the position, i is the dimension of the positional encoding, and d_{model} is the dimension of the model. The positional encoding is unique for each embedded token in the sequence, and in practice gives the model context for which tokens appear where in the input sequence.

The MHA sublayer consists of multiple attention heads. An illustration of a single attention head and the MHA sublayer can be seen in Figure 2.2.

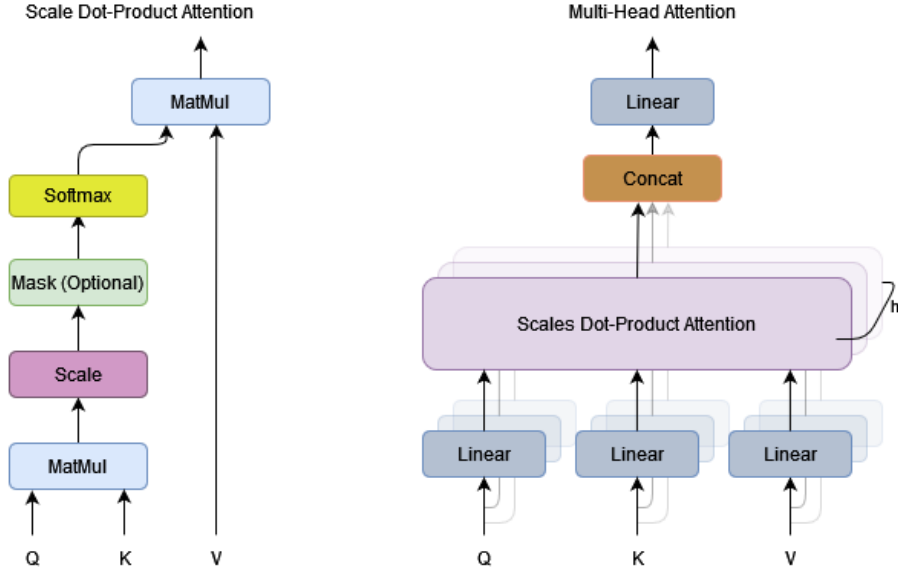


Figure 2.2: On the left, the scaled dot-product Attention is illustrated. On the right, the Multihead attention is illustrated. Here Q , K , and V are the query-matrix, key-matrix, and value-matrix, respectively.

In [3], the attention function implemented, in a single attention head, is called scaled dot-product attention and is calculated using (2.2).

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^\top}{\sqrt{d_k}} \right) V, \quad (2.2)$$

where d_k is the dimension of the queries and the keys, and the softmax function is calculated row-wise. The query-matrix (Q), key-matrix (K), and value-matrix (V) are created by row-wise concatenating the queries, keys, and values. This is done to speed up the computation.

When MHA is carried out the queries, keys, and values are projected by W_i^Q , W_i^K , and W_i^V , and then the calculations of the attention scores are carried out in parallel. This calculation is shown in (2.3).

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O, \\ \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \end{aligned} \quad (2.3)$$

where $W_i^Q, W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$.

Passing information only through the attention mechanisms of the transformer can lead to sub-par performance. In [9] it was shown that residual connections, as those illustrated in Figure 2.1, improve the accuracy of deep neural networks and make the networks easier to train. For this reason, residual connections are present in the transformer architecture presented in [3].

As mentioned in the beginning of this section, a decode layer consists of a masked MHA sublayer. The purpose of the masking is to stop leftward flow of information in the attention calculation.

2.2 Mistral 7B

Mistral AI is a French startup which quickly raised a lot of money in 2023 [10]. In September 2023 Mistral AI released their first LLM Mistral 7B under the Apache 2.0 license [11]. Within half a year they released the Mixtral 8x7B model under the same Apache 2.0 license, which outperforms similarly sized, or even some larger, models from both Meta and OpenAI [10]. Mistral AI’s CEO is quoted saying that Mistral AI is “[committed to pursuing] an open, responsible and decentralised approach to technology” [12]. The open nature and performance of the models released by Mistral AI make them good choices for this project.

The Mistral 7B model was first introduced in [8]. This architecture follows the one proposed in [3] closely, with four select modifications. The four optimizations implemented in [8] are Grouped Query Attention (GQA), Sliding Window Attention (SWA), Rolling Buffer Cache (RBC), and pre-fill and chunking.

GQA uses the same query for multiple attention calculations, which reduce the amount of data which has to be moved in and out of memory. SWA reduced the number of tokens a given attention calculation attends to by implementing a windowed approach. This window is slid over the tokens in sequence, to give the model the global context normal attention calculations would. RBC leverages the window size to reduce the amount of memory operations needed to carry out attention calculations. Pre-filling and chunking once again leverage the windowed approach of the model to reduce the number of memory operations needed for inference.

These modifications reduce the model’s memory footprint, increase inference speed, and allow for a larger context to be parsed by the model.

2.3 Prompting

This section is heavily based on the guide [13] and leans on the OpenAI Application Programming Interface (API) platform.

Interaction with an LLM is often done in a “chatting” context, where the user prompts the LLM, and a “conversation” can be had between the user and the LLM. What is of interest in this project is interaction with the intent of solving a task, such as music description. These tasks could also be text summarization, among many others.

How the prompt is written, or how large of a conversation history is passed to the model, can drastically change the output of the LLM. The process of creating different prompts and other interactions, such as allowing the LLM to query a database for information, is known as prompt engineering. In the following, we discuss a few of these techniques and

choose to only discuss methods within the scope of this project. Many more techniques exist, but they are beyond the scope of this project.

The most basic way of prompting an LLM is known as zero-shot prompting. A zero-shot prompt will not contain any examples which demonstrate how a task is successfully completed. An example of such a prompt is given in [13] and shown below.

Classify the text into neutral, negative or positive.
 Text: I think the vacation is okay.
 Sentiment:

In practice, this technique only works for basic tasks, and more advanced methods of prompting are needed for more complicated tasks.

Sometimes more context needs to be given to an LLM in order to solve a task. One way of providing this context is by using few-shot prompting. In few-shot prompting, examples are provided in the prompt itself, in order to condition the LLM on the knowledge present in the context. An example which showcases this "in-prompt" learning is given in [13] and shown below.

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:
 We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:

The provided example has 1 demonstration of what a task looks like, making it a 1-shot prompt. This number can be increased, mostly limited by the context size of the LLM used. This method does however not lend itself well to more complicated tasks.

The most complicated prompting technique we will explore in this report is chain-of-thought prompting, which was first introduced in [14]. This method of prompting in the most simple of cases asks the LLM to reason about its answer. This can be done in tandem with either zero-shot or few-shot prompting. As before zero-shot prompting provides no examples in the prompt itself, and few-shot prompting provides some examples. Examples of these techniques are provided in [15], and shown below.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
 A: Let's think step by step.

The above example showcases zero-shot chain-of-thought prompting where the answer is "guided" by the presence of *Let's think step by step*.

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can have 3 tennis balls. How many tennis balls does he have now?
 A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
 A:

The above example showcases few-shot chain-of-thought prompting, where a more detailed context is given to the LLM.

2.4 Evaluation

When training the model we can use a distance metric between the output produced by the model, and the ground truth. However, as mentioned in [2], several issues belie the task of evaluating a music captioning model.

One issue is that of semantic distance between summarizations or descriptions of music clips. Though the sentiment and quality of a description might match closely to another, the actual words used and in which order will harm the evaluation. These issues are addressed by several different evaluation models, such as ROUGE [16], BERTScore [17], or GPT-4 [18]. ROUGE uses an n-gram comparison approach, and as such it is susceptible to not matching synonyms. BERTScore uses learned embedding maps, in order to gauge semantic similarity without considering the words themselves. The GPT-4 based approach uses the large context window and good task solving skills of GPT-4 to produce evaluations which correlate strongly with human evaluations. These models show good performance in cases where the data only consists of text, such as summarizing articles, but as pointed out in [2] this does not solve all issues relevant to our project.

Another issue is that music description is inherently a creative task, where very different descriptions can be considered equally correct. This issue is solved in [2] by employing human evaluation, where humans are tasked with listening to a clip, and then judging two different descriptions of that music (one generated by the LLark model, and the other obtained from their music database). While we agree that this evaluation technique would theoretically produce the highest quality results, this is not feasible in this project due to time constraints.

Due to the time constraint of this project we choose to implement automatic evaluation. While a multi-modal model using GPT-4 multimodally where it has access to the music clips might produce the highest quality evaluations, this model would fall outside the scope of this project as well.

Instead, we choose to implement the BERTScore, since this is easy to implement and the latest models implementing the BERTScore show a Pearson correlation with human evaluation of approximately 0.6.

2.4.1 BERTScore

We employ a wrapper from Hugging Face of the open source implementation of the BERTScore, which is distributed under the MIT license. The code is written by the authors of [17], and can be found at [19].

Each evaluation by the BERTScore interface provides the recall, precision, and F1 score, which are performance metrics for binary classification problems. In these classification problems, the predictions and labels are either positive or negative. A true or false negative or positive is the case where the prediction either matches or does not match the label. The number of false negatives, true negatives, false positives, and true positives in the classification set are denoted FN , TN , FP , and TP , respectively. Using these quantities we can define

$$\begin{aligned} \text{Recall} &= \frac{TP}{P}, & \text{Precision} &= \frac{TP}{FP + TP}, \\ \text{F1 Score} &= \frac{2}{(\text{Recall})^{-1} + (\text{Precision})^{-1}}, \end{aligned} \tag{2.4}$$

where $P = TP + FN$ which is the number of positive labels. These metrics define the recall, precision, and F1 score in a binary classification problem [20].

The BERTScore metrics draw inspiration from (2.4), but are calculated differently. The general idea behind the BERTScore is that the captions and predictions are tokenized and then embedded by a given BERT model. The cosine similarity, which is the cosine of the angle between two vectors, is then computed between each pair of embedded tokens. These cosine similarities are then used to compute a sort of generalized recall, precision, and F1 score. A weighting is also applied to these scores based on the commonality of words. In practice, the scores provided by this method tend to fall in a small interval, instead of the theoretically possible interval of $[0, 1]$. To account for this a re-scaling can be carried out based on the empirically worst-case performance of the model. The empirical nature of the rescaling can result in values falling outside of the theoretical range of $[0, 1]$.

Example 2.1 illustrates how the BERTScore works for different sets of sentences. We used ChatGPT-3.5 from OpenAI to generate the sentences, and their rephrasings.

Example 2.1: BERTScore Example

In this example, only the F1 scores of the BERTScores will be used.

The BERTScore interface is loaded with the `microsoft/deberta-xlarge-mnli` model as this shows the best correlation to human evaluation amongst the available models. Baseline re-scaling is also enabled, and the model language is set to English.

The phrases used in this example are listed in Table 2.1.

F1 Score	Caption Sentences	Prediction Sentences
0.82	<i>The cat sat on the mat.</i>	<i>The feline rested on the rug.</i>
0.66	<i>She danced gracefully across the stage.</i>	<i>Her movements were elegant as she traversed the platform.</i>
0.26	<i>He wrote a heartfelt letter to his friend.</i>	<i>The aroma of freshly baked bread wafted from the bakery down the street</i>

Table 2.1: Example sentence pairs for the BERTScore calculation, along with their approximate F1 score.

It is not surprising the first sentence pair scores highly, since the prediction sentence is mostly constructed of synonyms of the words in the caption sentence. The score of the second sentence also makes sense, since they are semantically similar, but not necessarily describing the same thing. The most surprising thing about the last score is that it is not 0, since the sentences are not semantically similar, nor do they contain any significant amount of synonyms.

2.4. EVALUATION

This illustrates that the BERTScore can function as a rough guide for semantic similarity, but it is not sophisticated enough to deduce any meaning about a given sentence.

3 | LoRA and Quantization

In this chapter Low-rank Adapter (LoRA) [21] is introduced. LoRA is an approach developed for fine-tuning Large Language Models (LLMs). Hereafter, quantization theory for machine learning and QLoRA [22] are introduced. QLoRA is a quantized version of LoRA, which makes the process of fine-tuning LLMs more accessible since it requires less memory.

3.1 LoRA

LoRA is a method developed to fine-tune large-scale pre-trained language models. Existing techniques before LoRA often introduced increased inference time for a fine-tuned model. An approach that could combat this issue is to update all weights in the pre-trained model. However, this would be a very computationally heavy task for each time a model is fine-tuned. The advantages of using the LoRA approach are that it makes training more efficient, it does not introduce any additional computation time when running inference, and it makes it possible to use a pre-trained model to train multiple small LoRA modules each specialized for different tasks [21].

The general idea behind the LoRA approach is illustrated in Figure 3.1, where $W \in \mathbb{R}^{d \times k}$ represents the query, key, value, or output weight matrix. LoRA adds two new matrices, $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$ for each weight matrix, where $r \ll \min(k, d)$. LoRA then only updates A and B to calculate the new weights as

$$W' = W + \Delta W = W + BA, \quad (3.1)$$

where W' is the new weight matrix. A is initialized as entrance-wise standard normal distributions and B as a zero matrix. The output matrix h is then given as

$$b = Wx + \Delta Wx = Wx + BAx, \quad (3.2)$$

where x is the input. This means that at the first iteration, the contribution from A and B will simply be $b = Wx$ as $B = 0$, where after A and B are updated from the pre-trained weights.

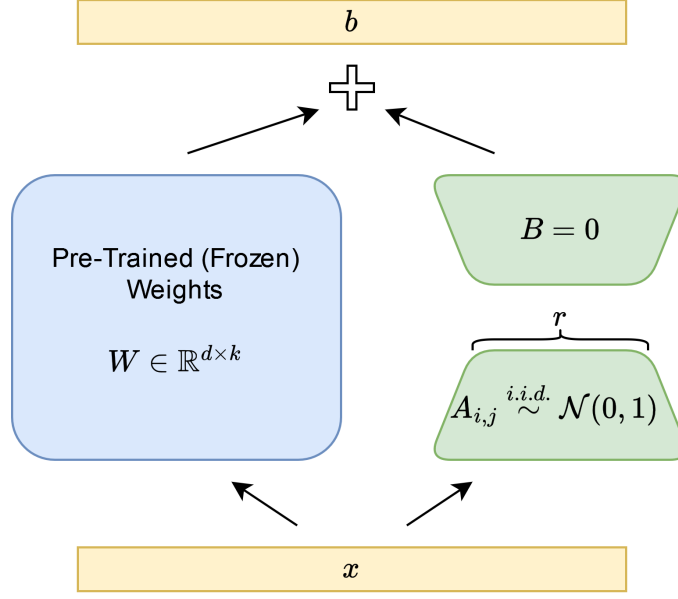


Figure 3.1: The reparametrization used in the Low-rank Adapter (LoRA) article. Only A and B are trained.

3.2 Quantization

Running LLMs can be a challenge on consumer hardware. For example, GPT-3 has around 176B parameters and just to run inference with a model of this size will take more than 700 GB memory [21].

Quantization is a technique used to reduce the computational and memory costs of running inference by representing the weights with low-precision data types like 8-bit integer (int8) instead of the usual 32-bit floating point (float32).

When quantizing from float32 to int8 it is common to rescale the input data type into the range of the target data type, using normalization by the absolute maximum value of the input elements. This is done because int8 only can represent 256 values and it is important to use the entire range of the int8 range [22]. Consider quantizing a float32 tensor ($X_{float32}$) into an int8 tensor (X_{int8}) of range $[-127, 127]$, we then get the quantisation

$$X_{int8} = \text{round}\left(\frac{127}{\text{absmax}(X_{float32})} \cdot X_{float32}\right), \quad (3.3)$$

$$= \text{round}(c_{float32} \cdot X_{float32}), \quad (3.4)$$

where $c_{float32}$ is the quantization constant. With this approach, large magnitude values, such as outliers, in the input tensor can cause problems. One problem is that some quantization bins will have very few or no numbers in them. In other fields, an easy solution to this could be discarding the outliers, but in machine learning the outliers are often the most important values [22].

Another common approach to prevent the outlier problem is to chunk the input tensor into blocks that are independently quantized with their own quantization constant. The input tensor $X \in \mathbb{R}^{b \times h}$ is chunked into n contiguous blocks of size B by flattening the input tensor and slicing it into n blocks, where $n = (b \cdot h)/B$. These blocks are then quantized

using (3.3), resulting in a quantized tensor and n quantization constants [22].

3.2.1 QLoRA

QLoRA is a method, which makes it possible to load and fine-tune a quantized 4-bit model without significant performance degradation compared to a 16-bit fully fine-tuned baseline model. Using QLoRA, a model with 65B parameters can be fine-tuned using less than 48 GB memory instead of more than 780 GB. To get this improvement in memory use, QLoRA introduces three innovations, namely: 4-bit NormalFloat, Double Quantization, and Paged Optimizers [22]. These innovations are combined to create a LoRA approach with a lower memory footprint.

4-bit NormalFloat

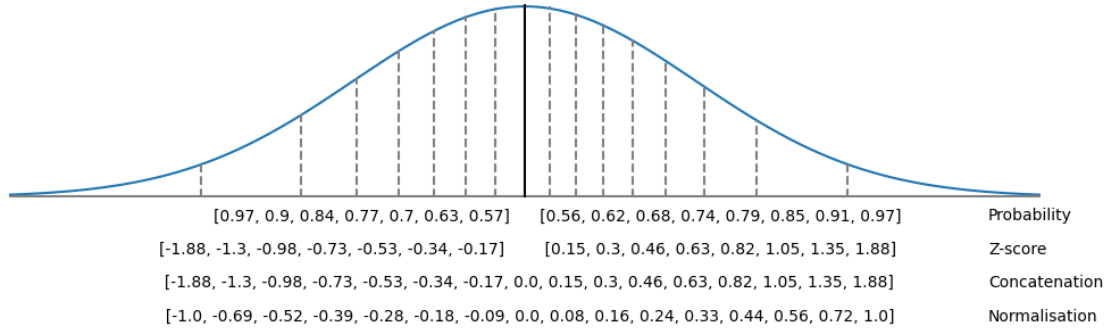


Figure 3.2: The idea behind 4-bit Normalfloat (NF4). The standard normal distribution is quantized into 17 quantiles, and the Z-scores are found, concatenated, and normalized.

The 4-bit Normalfloat (NF4) is a new datatype the authors of [22] have created to store data using less memory. NF4 is designed to be an ideal datatype for normal distributions, which makes sense since the weights of pre-trained neural networks usually follow a normal distribution with zero mean. The idea behind NF4 is illustrated in Figure 3.2. The weights of each chunk follow their own zero-mean normal distribution. The weights of each of these chunks are then scaled to fit into a chosen range. For NF4 the range is $[-1, 1]$ [22].

In the NF4 data type, the standard normal distribution is split into 17 quantiles, eight to the left of zero and nine to the right. This is done because we want an exact representation of zero. When the quantiles are found they are normalized to the range $[-1, 1]$. Now an input weight tensor can be quantized by normalizing it to the range $[-1, 1]$ using absolute maximum rescaling, which is equivalent to rescaling the standard deviation of the weight tensor to match the standard deviation of the data type [22].

Double Quantization

Double Quantization (DQ) is the process of quantizing the quantization constant. This can result in a memory saving of approximately 3 GB on a model with 65B parameters [22]. It is possible to get this kind of memory saving since there is an overhead when performing independent 4-bit quantization, on each block of a chunked input tensor. The overhead is caused by the n quantization constants, which usually are saved as float32. Quantizing all these using a single quantization constant is what results in the memory savings.

Paged Optimizers

Paged optimizers apply the NVIDIA unified memory feature, which automatically applies page-to-page transfers between the CPU and GPU to avoid errors when the GPU occasionally runs out of memory. The method works like memory paging between CPU RAM and disk. It is used to move the paged memory for the optimizer states to the CPU RAM from the GPU when it runs out of memory. When the memory is needed for the optimizer update step it is moved back to the GPU memory [22].

4 | Multimodal Models

Multimodal models refer to models capable of handling multiple modalities. The idea of combining multiple modalities is not new. However, recently there has been a significant increase in the study of multimodal LLMs [4]. The recent development in multimodal LLMs includes models that can handle modalities such as videos, images, audio, and text in a single model [23].

To create a multimodal model of any kind it requires a way to combine the different modalities into a shared space. In [5] they split these different ways into four categories, multimodal converters, multimodal perceivers, tools learning, and a data-driven approach. The multimodal converters take the output of one modality and input it directly to the LLM through a simple projection layer. The multimodal perceivers is used to minimize the semantic gap between the modalities by converting multimodal features to multimodal tokens that is consistent with the embedded representation of the LLM. The tools learning can be used as a way to extend the functionality of a given multimodal model by extending its capabilities with other foundation models. Lastly, there is the data-driven approach, which is often used to fine-tune a given multimodal model to fit a specific task. An example of this could be the LLava-Med [24], which is a LLava model [25] modified for medical data.

When working with audio as a modality, audio encoders are used to reduce the dimension of the audio while losing only a little information about the audio file. For combining an encoding and text, recent work has shown good performance of the multimodal converters, specifically a simple projection layer [25, 2]. The projection layer can consist of a simple linear layer, but it may also contain multiple trainable layers. The most important part of the projection is that it manipulates the size of the encoded audio to match the input size of the LLM, without excessively reducing the amount of information in the embedding. A common issue with the projection layer approach is hallucination, which may arise from misalignment of the encoding and text features. There is a two-stage solution that is often used to combat this issue. Firstly, the projection layer should be trained while the parameters of the LLM and the encoder are frozen. Secondly, the trained projection layer should be trained together with the LLM with only the parameters of the encoder being frozen [5, 25].

When using the model setup suggested in Figure 1.3, both the encoder, the LLM, and the audio projector are trainable. Here it is possible to use an off-the-shelf audio encoder and a pre-trained or fine-tuned, LLM. In [2] they use an existing audio encoder, and a pre-trained LLM.

4.1 Multi-Window Masked Autoencoder

The encoder used in this project is part of the Masked Autoencoder (MAE) family of encoders. These encoders work by passing only a few patches of an image, encoding these patches, and passing the encoded patches to the decoder [26]. In order to apply such an encoder on an audio signal a spectrogram is created based on the audio.

A model that uses the MAE structure is the Multi-Window Masked Autoencoder (MW-MAE) [27]. The model structure of MW-MAE is illustrated in Figure 4.1. They use log mel-spectrograms as input and partition them into non-overlapping patches. In our model, we only use the encoder of the MW-MAE and hence the decoder is not discussed further in this report.

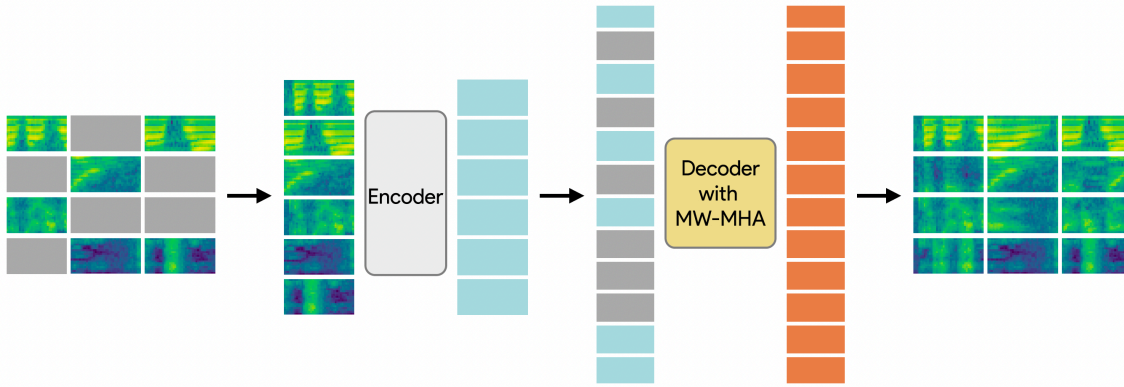


Figure 4.1: Overview of the Multi-Window Masked Autoencoder (MW-MAE) architecture. Image is from [27].

The encoder uses 20% non-masked patches which are chosen randomly. Hence, the majority of the patches are not processed during the training of the encoder. The positional embedding of the visible patches is added using a sinusoidal positional embedding.

4.2 J2A-2.x Implementation and Architecture

This section describes the design and implementation of two multimodal models, as well as an approach to training the models. The two models are called J2A-2.0 and J2A-2.1, together referred to as J2A-2.x. The implementations are inspired by the approach used in [28]. This approach utilizes `Mistral-7B-OpenOrca`¹ and follows the multimodal conversion approach. Furthermore, we also elaborate on different specifications and decisions that have been made in the process of creating the two models. To test the Hugging Face framework and to get familiar with the necessary tools for this project, we have also created a simple model that is found in Appendix A but will not be considered further in this report.

The two models are differentiated by sub-version instead of major version since they are architecturally very similar. These models are multimodal models and can parse music clips to obtain a text description. This is enabled by employing the Multi-Window Masked Autoencoder (MW-MAE) [27] and projecting the encoded audio to the input layer of the

¹<https://huggingface.co/Open-Orca/Mistral-7B-OpenOrca>

underlying Large Language Model (LLM). Figure 4.2 illustrates the architecture for J2A-2.x.

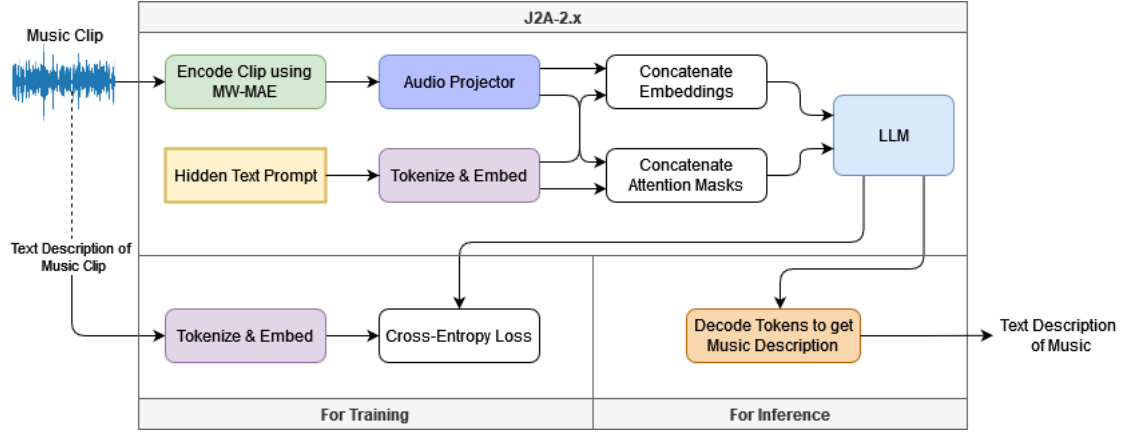


Figure 4.2: A flowchart overview of the J2A-2.x model.

In Figure 4.2 it can be seen that both for training and inference a hidden hard-coded single prompt is parsed to the LLM alongside the encoded music clip. This is done to guide the model in what we would like it to focus on and to understand the task. For training the output of the LLM, it is compared to a tokenized label, which is seen as ground truth. The comparison is done using cross-entropy loss. The loss is then used to update the parameters in the model. For inference, the output of the LLM is detokenized to get the output in a text format. This text is then the description of the music.

The data used to train the models comes from two datasets, the YouTube8M-MusicTextClips [6] and the MusicCaps [7] datasets. YouTube8M-MusicTextClips consists of over 4 thousand songs, each with a description made by the YouTube video annotation system. MusicCaps has over 5 thousand songs with a similar description but written by human experts. Furthermore, MusicCaps also has a list of music aspects such as genre, mood, instrumentation, etc.

The text captions are around four sentences long in both datasets. The descriptions are based on 10-second clips of the YouTube videos. Combined, the two datasets yield a total of about 9 thousand songs, as some of the videos were unavailable as they were marked as unavailable, private, or age-restricted.

The models are not trained, tested, or validated on the exact same training, test, and validation datasets. However, they use the same number of songs for each of the three categories. We combined and shuffled the data from MusicCaps and YouTube8M-MusicTextClips and used 80% for training, 10% for validation, and 10% for testing. The size of the different datasets is shown in Table 4.1. In Table B.1 more training configurations are specified.

The output of the encoder is a tensor in $\mathbb{R}^{2 \times 690 \times 3840}$. This tensor needs to be mapped to \mathbb{R}^{4096} in order to make it match the input layer of the LLM. The difference between J2A-2.0 and J2A-2.1 is in how this task is performed.

J2A-2.0 does so by considering the tensor as a set of 690 matrices and takes the mean of these in order to get a matrix in $\mathbb{R}^{2 \times 3840}$. This matrix is then passed to the audio projector in Figure 4.3a. From previous versions of J2A-2.0 we suspected that the encoding did not contain enough information for our model and therefore we created J2A-2.1.

	Train	Test	Validation
Size	7212	902	902
Hours	20.0	2.5	2.5

Table 4.1: Size of the train, test, and validation dataset.

J2A-2.1 instead considers the tensor to be two matrices in $\mathbb{R}^{690 \times 3480}$. An average is then taken over these 2 matrices in order to get a single matrix in $\mathbb{R}^{690 \times 3480}$. This matrix is passed to the audio projector in Figure 4.3b where a pooling layer is included before a normalization layer and the linear layer.

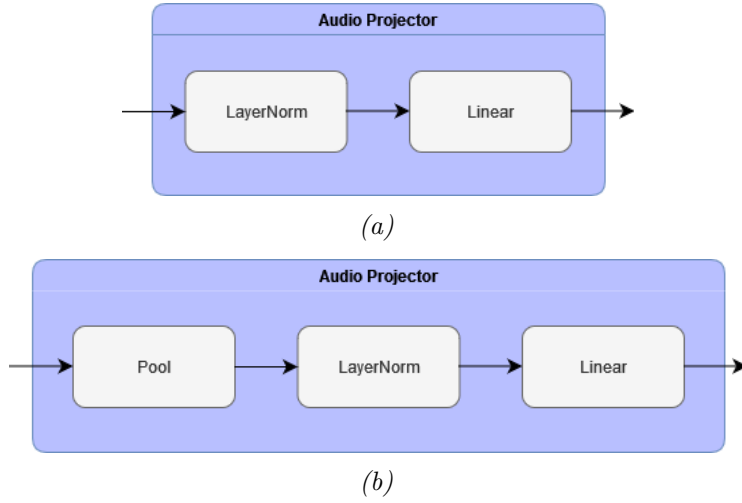


Figure 4.3: An insight of how the Audio Projector from Figure 4.2 looks for J2A-2.0 in Figure 4.3a. Similarly, the audio projector for J2A-2.1 is shown in Figure 4.3b.

The hard-coded single prompt for J2A-2.x consists of a system message to determine how the language model should respond. Then there is a user message that specifies what the model should find an answer to. The prompt can be found in Listing C.1. The system and user persona method of witting prompts is inspired by the OpenAI API.

In this project, we use the Llama tokenizer². This tokenizer is based on an unsupervised text tokenizer and detokenizer called SentencePiece. This is a fast C++ based subword tokenizer that is trained directly on sentences that are not pre-processed [29]. The tokenizer takes frequent word combinations on the data it is trained on and encodes this together, e.g., "ll" could be a common character combination that is saved as one token.

After the tokenization, the embedding works as a lookup table for a token to a vector that carries more information about each token. The different Mistral models use `torch.nn.Embedding`, which is a matrix with the vocabulary size of the tokens as the number of rows and the hidden size of the language model as the number of columns.

For training, only the gradients of the audio projector are updated, hence the audio encoder and LLM remain the same throughout training. Since the LLM is not being fine-tuned

²https://huggingface.co/docs/transformers/main/en/model_doc/llama

during training we have used the fine-tuned version of **Mistral-7B-Instruct-v0.2** called **Mistral-7B-OpenOrca**, which is trained to be specialized for chatting.

Before presenting the model tests we will look into the loss curves, shown in Figure 4.4. For both models, the loss curve from the training data follows the one calculated on the evaluation data. This indicates that the models are not overfitted. The loss curves also indicate that it is not necessary to train the models more than ~ 50 epochs. The loss curves for both models seem similar after the initial drop that happens within the first few epochs. Both models have been trained on an NVIDIA L40 GPU with 48 GB memory. The training time for **J2A-2.0** is 114,6 hours and 141 hours for **J2A-2.1**. The difference in training time is probably because of the larger audio projection layer in **J2A-2.1**.

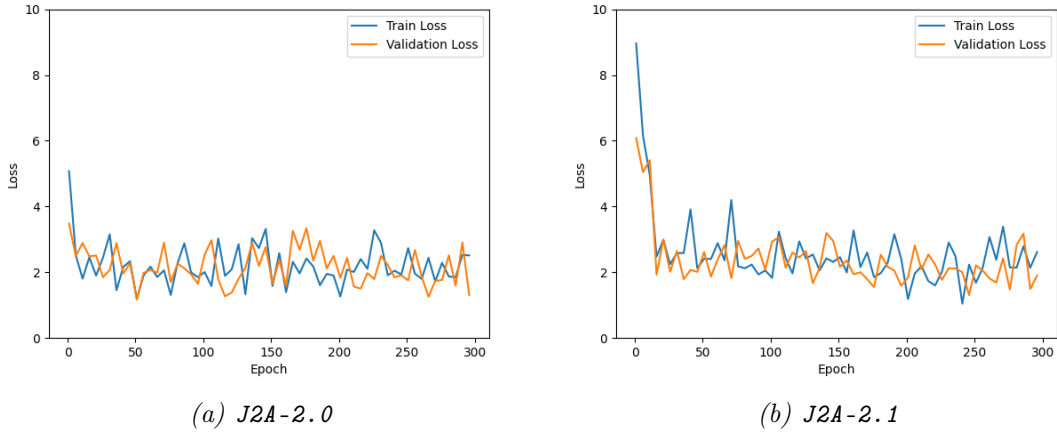


Figure 4.4: Training and validation loss.

5 | Experiment

In this chapter, we present the results from testing J2A-2.0 and J2A-2.1 that are described in Section 4.2. We will use the BERTScore that is elaborated on in Chapter 2, to make a quantitative evaluation of the text that is obtained using the two models.

The BERTScore can be interpreted as a lower bound for the model performance in the sense that the labels may not always encapsulate all information about a given song. Hence, a song may get a bad BERTScore if it says something different than the label, even if it describes the music well.

We also conduct a qualitative analysis of chosen predictions, where we comment on the prediction versus the labels and the music clip itself.

5.1 Quantitative Evaluation

Our evaluation set consists of 902 music clips. We calculate the BERTScore for each of the inferences calculated by the models. This generates 2 sets we can use for evaluation of the models.

$$E_{2.0} = \left\{ L_i, P_i^{2.0}, F(L_i, P_i^{2.0}) \mid i \in [1, 902] \right\}, \quad (5.1)$$

$$E_{2.1} = \left\{ L_i, P_i^{2.1}, F(L_i, P_i^{2.1}) \mid i \in [1, 902] \right\}, \quad (5.2)$$

where L_i and P_i are the label and prediction, respectively, for music clip i , and F is the F1 score of a label and prediction.

After calculating the BERTScore of each label-prediction pair, we will inspect the distribution of F1 scores over the evaluation data.

In order to get an overview of the general performance of the models a histogram of their F1 scores has been created. These histograms can be seen in Figure 5.1.

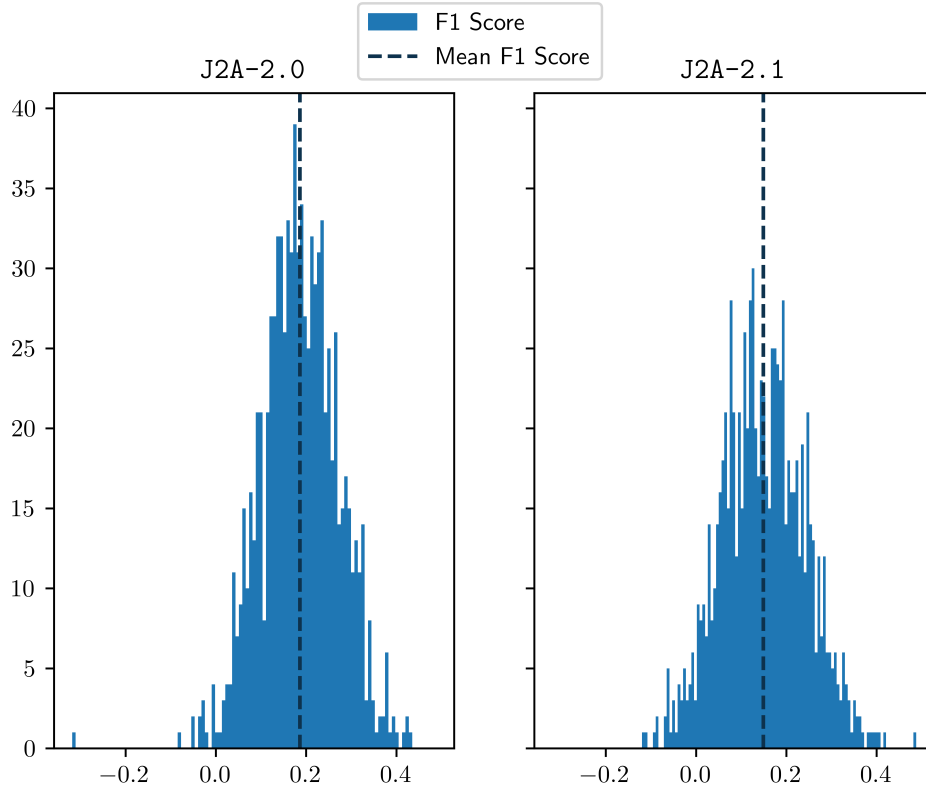


Figure 5.1: Distribution of F1 scores of inferences carried out in evaluation of J2A-2.0 and J2A-2.1.

As mentioned in Section 2.4.1 the F1 score is a scalar in the closed interval $[0, 1]$. However, because of the re-scaling they can fall below zero, when the score is close to zero before re-scaling, as is the case in Figure 5.1.

	J2A-2.0	J2A-2.1
Mean	0.186	0.149
Variance	0.007	0.009

Table 5.1: Mean and variance of the F1 scores for J2A-2.0 and J2A-2.1.

The results in Table 5.1 do seem to imply that the J2A-2.0 outperforms J2A-2.1. This will be discussed further in Chapter 6.

To visualize how close the distributions of the F1 scores are to each other a QQ-plot is illustrated in Figure 5.2. The QQ-plot shows that the distributions are quite similar, except for a few outliers. The largest outlier in Figure 5.2 corresponds to the response in Table 5.2.

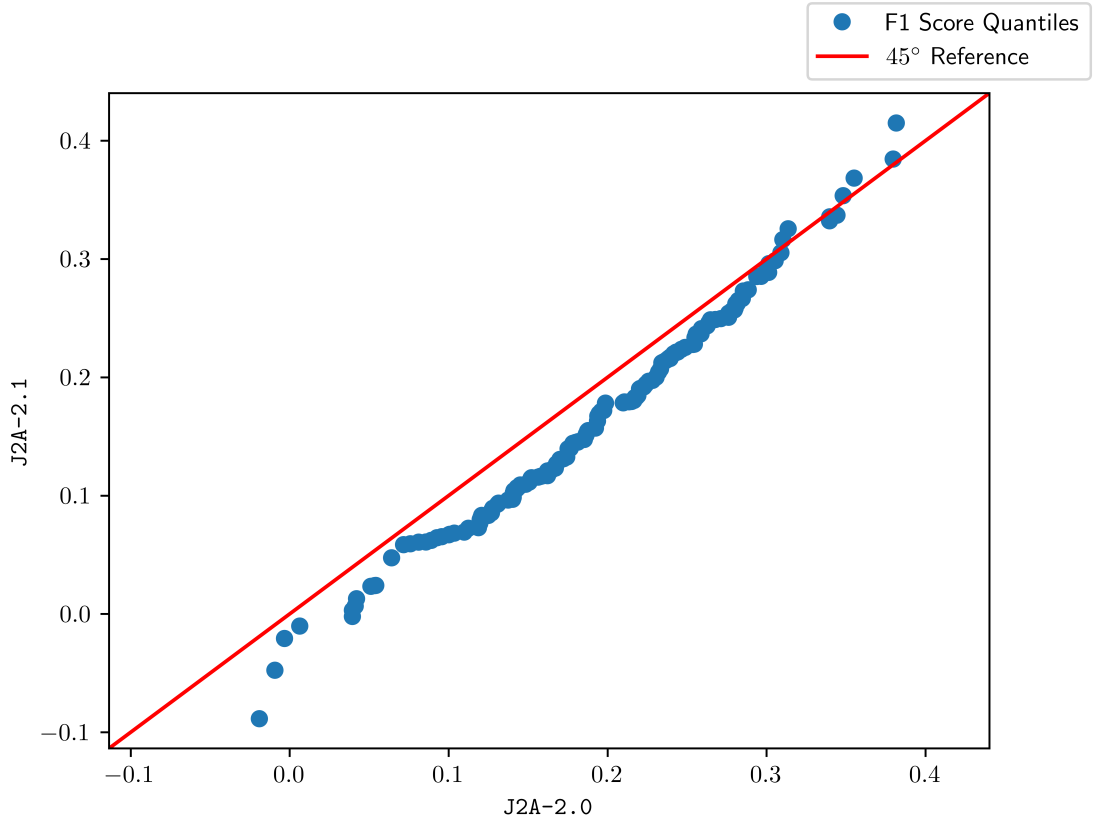


Figure 5.2: QQ of J2A-2.0 and J2A-2.1.

An interesting test is to examine the correlation between the F1 scores. For this the Pearson correlation [30] has been used. Only 161 data points are in common between the test dataset of J2A-2.0 and J2A-2.1. These are the only points used to calculate the Pearson correlation. The Pearson correlation is 0.382. This means that there is some correlation between the F1 score of the two models and the song that is scored. Although the score may not seem convincingly high, it should be noted that running inference twice on the same model would also result in varying F1 scores and may therefore result in a similar Pearson correlation.

5.2 Qualitative Evaluation

This subsection is a subjective look into the results produced by J2A-2.0 and J2A-2.1. A subset of the inferences will be evaluated qualitatively, in order to get a more nuanced understanding of the models performance than what the BERTScore can provide. This section includes the inferences with the best and worst F1 scores, along with 2 randomly picked examples.

An index of what clip belongs to which inference table can be found in Table 5.8. The table includes external links to the videos on YouTube where the clips are from. The links are timestamped and the clips are the 10 seconds following that timestamp.

In Table 5.2 the worst response for J2A-2.0 is presented. Here it is clear that the response from J2A-2.0 is bad, since it response with a combination of Cyrillic and Chinese

characters.

Label	J2A-2.0 -0.32
This slow pop song features a male voice singing the main melody. The voice is emotional. This is accompanied by a piano playing chords in the background. A synth plays a repetitive two note phrase in this song. There is no percussion in this song. The mood of this song is romantic. There are no other instruments in this song. This song can be played in a dreamy sequence in a romantic movie.	散落月光的 Я янва 罗瑕, 同时仿佛遭怕成風 thing. слаусс. 愛F, 近 lum 楊阿: target for 號 her 細

Table 5.2: Label and inferences for J2A-2.0, where J2A-2.0 scored the worst. The F1 score is written in the top right.

In Table 5.3 the worst response for J2A-2.1 is presented. Here it can be seen that the response from J2A-2.1 has multiple short sentences, which are music-related, but they are contradicting. For example lounge pop and pop rock would not be the same. Furthermore, it repeats itself in two ways. One way it repeats itself is simply having the same sentence twice, here it is "Great for pop song". Another way is that it mentions a lot of different sub-genres of pop.

Label	J2A-2.1 -0.12
This song contains an acoustic drum-set holding the groove and changing his beat after a fill-in to a four on the floor with a kick on every beat. Mixed voices are singing a soulful sounding melody then an e-piano takes over playing a melody before a trumpet and a saxophone come in playing a loud melody. In the background you can hear percussion such as a cowbell being panned to the right side of the speakers. The crowd is cheering and clapping along. This song may be playing at a live concert.	Eastern-sounding music. Great for pop song. Great for pop song. Perfect for radio commercial. Shinkicker. Euro pop lounge song. Pop dance floor track from the east. Lounge pop. Pop rock song. No

Table 5.3: Label and inferences for J2A-2.1, where J2A-2.1 scored the worst. The F1 score is written in the top right.

In Table 5.4 the best response for J2A-2.0 is presented. Here it can be seen that the response from J2A-2.0 is coherent and gets multiple things correct, such as male vocal

5.2. QUALITATIVE EVALUATION

and the fact that it is hip hop. Also, there are no statements which are plain wrong, although the last sentence does not make sense.

Label	J2A-2.0	0.44
Loopy hip hop track with laid back male vocal with modulated electronic elements having a spooky vibe.	Lush, clean-sounding hip-hop track with four different layers of autotuned male vocals and lo-fi production. Has a moody vibe. Could easily be paired as part of an "artifact" intro prom	

Table 5.4: Label and inferences for J2A-2.0, where J2A-2.0 scored the best. The F1 score is written in the top right.

In Table 5.5 the best response for J2A-2.1 is presented. Here it can be seen that the response is correct in multiple aspects, such as the gender of the singer, some instruments, and genres are very similar. It also mentions two artists who are both rappers. The response says that one of the artists is lip syncing, which is incorrect and an example of the model hallucinating.

Label	J2A-2.1	0.49
Funky rap song with male vocals, electronic trap beats and bass drops. Great for party mood.	groovy hip hop track with male rap vocal, grooving bass and positive synth with snared drumming. Sounds party. Excellent for dance clubs. A\$AP Rock featuring 21 Savage lip syncing.	

Table 5.5: Label and inferences for J2A-2.1, where J2A-2.1 scored the best. The F1 score is written in the top right.

Two examples of the results are presented in Table 5.6 and Table 5.7 together with three examples in Appendix D. Looking at the response from J2A-2.0 in Table 5.7 it can be seen that the response includes that the music clip contains drumming, which is correct, but not mentioned in the label. Here we also see an example of the model being completely wrong, since the song is not sung by a K-pop band.

It can be seen that the responses from both models are often coherent and make sense at the beginning, but become more incoherent or wrong, as you read through them. Some of the things that the models often mentions are the vocal, the genre, and some of the instruments. In general the models often include something correct, but also something incorrect. It also seems like the two models have similar F1 scores for same clips, which does not contradict the calculated Pearson correlation calculated in section 5.1. Here Table 5.6 is an example where the F1 scores are further apart.

Label	J2A-2.0 0.22	J2A-2.1 0.08
This is a rock and roll piece used as an accompaniment for a dance. There is a male vocal singing at a low-to-medium pitch. The main melody is being played by an electric guitar while the bass guitar is repeating a bass line with a groovy pattern in the background. The acoustic drums are playing an upbeat 4/4 rock and roll beat. There is a raw feeling to it. The tune is catchy. This piece could be playing in the background of a sports bar or a rock bar. It could also work well at sports venues.	Pop track with female vocals, groovy guitar lines and percussive elements. Great for breakdancing. Doing the dance. Groovy in nature. Is cheerful and somewhat composition. Has an up beat vibe to it.	Pop song youth vibes with processed male vocals, drum beats and bass lines along with beautiful synthDMA. Great for standing on stage. Straight-forward pop with impact. Perfect. Männer Bugle by Helmau

Table 5.6: Random sample of label and inferences. The F1 score is written in the top right.

Label	J2A-2.0 0.15	J2A-2.1 0.12
This is an upbeat, soulful and funky song. It's got a catchy and fast paced rhythm which makes it good to dance to. The singer has a raspy and energetic voice.	Pop song with boyish high pitched male vocals and drumming tempo. The track feels closely resembles untalented lyrics being sung by K-pop boy band. Feels soothing with minimal bass, however the chorus is p	Hoovering sample featuring trap beats and akin pads and various taps. Ideal for underground rapper expression. Ideal for underground rapper. Quality is poor. the zona sound is well versed in this Latin tune.

Table 5.7: Random sample of label and inferences. The F1 score is written in the top right.

Database ID	Table reference	URL
Table 5.2	MusicCaps_680	https://youtu.be/5h5NdW6cYY0?t=30
Table 5.3	MusicCaps_2945	https://youtu.be/UoxHw0l2gN0?t=10
Table 5.4	YouTube8M-MusicTextClips_2643	https://youtu.be/nSKbB0FsPg4?t=87
Table 5.5	YouTube8M-MusicTextClips_324	https://youtu.be/YH_RsDHvw0w?t=100
Table 5.6	MusicCaps_1441	https://youtu.be/D8-x1T8M4gk?t=30
Table 5.7	MusicCaps_4059	https://youtu.be/grE0lwTsSPg?t=30
Table D.1	YouTube8M-MusicTextClips_3860	https://youtu.be/falPDZ5xuMo?t=125
Table D.2	MusicCaps_781	https://youtu.be/6jeq5lP5Up0?t=30
Table D.3	YouTube8M-MusicTextClips_1094	https://youtu.be/7J9AXxZB0R8?t=125

Table 5.8: Table linking the inference tables to their respective music clips.

6 | Discussion

Two models have been created using the multimodal converter structure, those being J2A-2.0 and J2A-2.1. We chose this method as it provides a simple way to pass the audio to the Large Language Model (LLM) and others have had great success using this approach [2].

An important part of the implementation is that only the parameters in the projection layer are updated during training. As mentioned in Chapter 4 this is only the first step of training a multimodal converter model, where the second step is fine-tuning both the LLM and the projection layer together. We have not implemented the second step of this process due to time constraints.

The quantitative evaluation in Chapter 5 shows that J2A-2.0 performs marginally better than J2A-2.1 since it has a slightly higher mean and a smaller variance, which indicates that J2A-2.0 is more consistent in the quality of its responses. Both models generally perform poorly, their means are both lower than the bad example in Example 2.1, where the F1 score was 0.26. As mentioned in the beginning of Chapter 5 the F1 score can be interpreted as a lower bound for the model performance. Therefore, the low F1 score does not guarantee poor performance.

In the qualitative evaluation, we see that the models' responses are often about music, and sometimes in coherent sentences, which indicates that the models are aware of the subject of the music captioning task. This can also be because the LLM knows this from the prompt since we specify that it should describe music. Even though the responses contain music descriptions, the model still seems to hallucinate a lot. There are multiple examples where the response is wrong on both genre, instruments, and gender of the singer. This is likely because only the projection layer of the model is trained [5]. Although the sentences are sometimes coherent, part of the responses is often gibberish. They often start with descriptions that make sense and usually match the music clip. Towards the end of the responses, the quality quickly diminishes. This can be because the LLM itself needs more training on music-related data or for the specific task at hand.

The prompt used for the two models, shown in Listing C.1, specifies that the model prediction should not be repetitive. Although this is specified, we still encounter this type of repetition in the case shown in Table 5.3. We also specify that it should describe the music style and any unique features of the clip, its chords, tempo, and the instruments used. Although this is specified the model ends up often mentioning the vocal, the genre, and some of the instruments. This may be due to the labels not containing any information about chords and tempo. Therefore, the model does not get a lower loss if these things are mentioned. This might indicate that the prompt should be changed. We could either make the prompt more simple or make sure to ask for things that are also mentioned in

the captions.

In general, there is no noticeable performance difference between the two models. The only clear difference is the F1 score, where the J2A-v2.0 seems to perform a little better, but this is not enough to conclude that one model is better than the other. The QQ-plot and Pearson correlation indicate that the larger audio encodings and the extra layer in the audio projection do not yield any noticeable improvements for J2A-2.1 as compared to J2A-2.0.

There is a large variation in quality and content in the data. It contains everything from rap music to guitar lessons¹. This indicates that the quality of data is not the best for our use case, as we are only interested in captioning music clips. If there are a limited amount of these types of clips then it may not be enough to have a large impact on the models. The quality of the recordings is also not consistent. There are examples of both live recordings with audience noise and older recordings of lower quality. This might make the models more resilient to noise in the data, but this is not within the scope of this project. As such, it would be preferable to have only high quality recordings in the data for this project.

Our proposed models might produce worse results than LLark [2] in their current state. However, our models also have several advantages. The encoder and LLM used in our models are distributed under the Apache 2.0 license. The MusicCaps dataset is distributed under a Creative Commons license, and the YouTube8M-MusicTextClips dataset is distributed under a permissive research license from Adobe. This means that the J2A-2.x models are open and free to use for academic and teaching purposes. If the YouTube8M-MusicTextClips dataset was not used during training, then the J2A-2.x models would fall under the Apache 2.0 license, allowing commercial uses. Since our model is more focused on its music captioning task than the more general set of tasks of LLark, we expect J2A-2.x to be more computationally effective if current issues are fixed.

6.1 Further Research

As mentioned in Chapter 4, a common way to combat the hallucination issue is to fine-tune the projection layer together with the LLM. This approach is the highest priority for further research, as training only the projection layer rarely yields the best performance for a multimodal converter model.

Further research might also include using a larger and better performing LLM. An example of another language model that could be tested is Mixtral-8x7B [31].

In this project, we implemented only zero-shot prompting. Using different techniques, such as zero- to many-shot chain-of-thought prompting might produce better results, than what we were able to achieve. Doing so would require one to consider the multi-modality of the models, and how one might be able to integrate examples of the projected music clip, in order to gain better performance by changing just the prompt. Another approach could also be to test a more simple prompt where the user does not ask for it to describe specific things about the music clip, but rather to make a general music description. Lastly, it could also be of interest to train a model using no prompt to see how the performance is affected. This is mostly of interest when also training the LLM since this might learn to

¹An example of a guitar lesson is <https://www.youtube.com/watch?v=IqGB4nQIAcQ&t=210s> from the MusicCaps dataset.

answer in the music domain from the labels only. A challenge with this is that it might require more data.

Since the Multi-Window Masked Autoencoder (MW-MAE) is only used for encoding, it might be of interest to test how using other audio encoders might impact the performance of the models. Encoders that could be interesting to test are the Jukebox encoder [32] since it is used for the LLark model [2] and the Whisper encoder [33], since it is used in [28].

When testing the models it would be interesting to look at some of the metadata in the MusicCaps [7] dataset. Specifically, the dataset contains an aspect list with relevant keywords for each song in the dataset. This could for instance be 'male voice' or 'instrumental music'. It would be interesting to check through each of these lists for each song in the test data set and see if our model response contains any of these keywords or perhaps synonyms of the keywords. Since we are interested in how well we can describe a given music clip, this approach may provide an alternative to only looking at the F1 score [19].

Another approach for this project would have been trying different ways of combining the modalities. As discussed in Chapter 4 there are multiple ways to combine different modalities [5]. In this report, we use the multimodal conversion approach. However, a similar approach could be to use a multimodal perceiver which may give the LLM a better context for the audio encodings. This approach can be thought of as an initial conversion of audio to text before the LLM processes the audio encoding. The tools learning approach may also be a good alternative. With this approach, one can use an existing model and extend its features in a toolbox-like manner. This can be done by using an existing image-to-text model and modifying it to handle the audio modality, or reshaping the encoding such that it can be parsed as an image. The advantage of this approach is that the language model is already fine-tuned for a caption task, and only needs to be fine-tuned to describe music clips. The data-driven approach is also a feasible option and can be employed by fine-tuning existing audio captioning models for our more specific music captioning task. This could for instance be implemented by using an audio captioning model like ACTUAL [34] or the one proposed in [35].

7 | Conclusion

In this project, the goal was to create a multimodal model that is able to describe 10 second music clips. To do so we chose to build a model with the multimodal converter approach, using a projection layer to pass the audio to the Large Language Model (LLM). We chose this method as it provides a simple way to pass the audio to the LLM and others have had great success using this approach. We developed and evaluated two model variants, J2A-2.0 and J2A-2.1. We did this to test if a more complex projection layer would increase the performance of the model, J2A-2.1 being the one with a more complex projection layer. While J2A-2.0 showed slightly better performance in the quantitative evaluation, both models generally struggled, particularly with coherence and specificity in their responses. In conclusion, the overall results highlighted significant challenges that need addressing, including hallucination and inaccurate statements for both models.

To address these challenges the trained audio projection layer should be trained together with the LLM. Moreover, further research into other LLMs, trying other prompt strategies, and audio encoders, in order to improve the models' performance is of interest.

Bibliography

- [1] Mark Savage. The global music market was worth \$26bn in 2021. *BBC*, 03 2022. Available at: <https://www.bbc.com/news/entertainment-arts-60837880> (Accessed 06/05/2024).
- [2] Josh Gardner, Simon Durand, Daniel Stoller, and Rachel M. Bittner. Llark: A multimodal instruction-following language model for music, 2024.
- [3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [4] Jiayang Wu, Wensheng Gan, Zefeng Chen, Shicheng Wan, and Philip S. Yu. Multi-modal large language models: A survey, 2023.
- [5] Shezheng Song, Xiaopeng Li, Shasha Li, Shan Zhao, Jie Yu, Jun Ma, Xiaoguang Mao, and Weimin Zhang. How to bridge the gap between modalities: A comprehensive survey on multimodal large language model, 2023.
- [6] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark, 2016.
- [7] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. Musiclm: Generating music from text, 2023.
- [8] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Deven-dra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [10] Vishal Mathur. A french ai start-up may have commenced an ai revolution, silently, 2023.
- [11] Mistral AI team. Bringing open ai models to the frontier, 2023.
- [12] Lucas Ropek. Buzzy startup just dumps ai model that beats gpt-3.5 into a torrent link, 2023.

- [13] Elvis Saravia. Prompt Engineering Guide. <https://github.com/dair-ai/Prompt-Engineering-Guide>, 12 2022.
- [14] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [15] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.
- [16] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.
- [17] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.
- [18] Liyan Tang, Zhaoyi Sun, Betina Idnay, Jordan G Nestor, Ali Soroush, Pierre A Elias, Ziyang Xu, Ying Ding, Greg Durrett, Justin F Rousseau, et al. Evaluating large language models on medical evidence summarization. *npj Digital Medicine*, 6(1):158, 2023.
- [19] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. Available at: https://github.com/Tiiiger/bert_score.
- [20] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.
- [21] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- [22] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.
- [23] Chenyang Lyu, Minghao Wu, Longyue Wang, Xinting Huang, Bingshuai Liu, Zefeng Du, Shuming Shi, and Zhaopeng Tu. Macaw-llm: Multi-modal language modeling with image, audio, video, and text integration. *arXiv preprint arXiv:2306.09093*, 2023.
- [24] Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. Llava-med: Training a large language-and-vision assistant for biomedicine in one day, 2023.
- [25] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.
- [26] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021.

-
- [27] Sarthak Yadav, Sergios Theodoridis, Lars Kai Hansen, and Zheng-Hua Tan. Masked autoencoders with multi-window local-global attention are better audio learners, 2023. ICLR2024, The Twelfth International Conference on Learning Representations, Vienna, Austria.
- [28] Paul Mou. Listening with LLM. Available at: <https://paul.mou.dev/posts/2023-12-31-listening-with-llm/>.
- [29] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.
- [30] Israel Cohen, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. *Noise reduction in speech processing*, pages 1–4, 2009.
- [31] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L  lio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Th  ophile Gervet, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. Mixtral of experts, 2024.
- [32] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.
- [33] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022.
- [34] Yiming Zhang, Hong Yu, Ruoyi Du, Zheng-Hua Tan, Wenwu Wang, Zhanyu Ma, and Yuan Dong. Actual: Audio captioning with caption feature space regularization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.
- [35] Minkyu Kim, Kim Sung-Bin, and Tae-Hyun Oh. Prefix tuning for automated audio captioning. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.
- [36] Sebastian B  ck, Filip Korzeniowski, Jan Schl  ter, Florian Krebs, and Gerhard Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 10 2016.
- [37] pedromatias97. music-genre-detector-finetuned-gtzan_dset. Available at: https://huggingface.co/pedromatias97/genre-recognizer-finetuned-gtzan_dset.

Nomenclature

API	Application Programming Interface
BPM	Beats Per Minute
DNN	Deep Neural Network
DQ	Double Quantization
FNN	Feedforward Neural Network
GQA	Grouped Query Attention
GRNN	Gated Recurrent Neural Network
LLM	Large Language Model
LSTM	Long Short-Term Memory
LoRA	Low-rank Adapter
MAE	Masked Autoencoder
MHA	Multi-Head Attention
MW-MAE	Multi-Window Masked Autoencoder
NF4	4-bit Normalfloat
NLP	Natural Language Processing
RBC	Rolling Buffer Cache
RNN	Recurrent Neural Network
SWA	Sliding Window Attention
float32	32-bit floating point
int8	8-bit integer

A | J2A-1.0

This model is constructed from fine-tuning the `Mistral-7B-Instruct-v0.2` model. The prompt used for this model is heavily inspired by the example given in [2] from their captioning task. The features, described below, are merged with the prompt to give context to the LLM. An overview of the model is seen in Figure A.1.

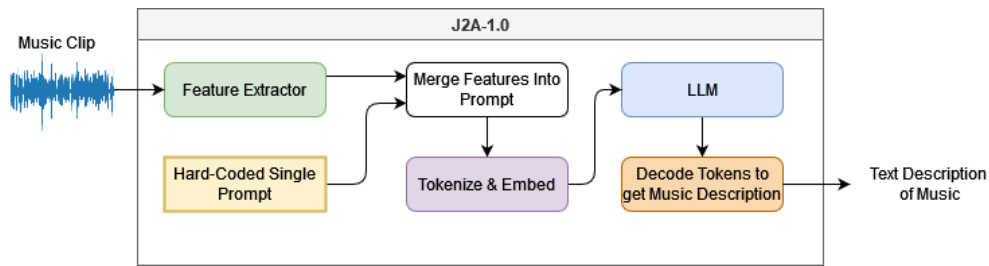


Figure A.1: A flowchart overview of the J2A-1.0 model together with the data that has been used to train the model.

A.1 Features

From the songs, we extracted features inspired by the approach in [2]. We use [36] to extract Beats Per Minute (BPM), key, downbeats, and chords. We also use [37] to generate genres for the songs.

The chords and downbeats are given in inequidistant time intervals where a new interval is started each time the chord or downbeat in the audio changes. The genre classifier has five overall categories (reggae, hip-hop, rock, and country) and comes with a percentage indication of the likelihood that a song is a given genre.

B | Training Configurations

Specification	
Learning rate	0.0015
Optimizer	<code>torch.optim.Adam</code>
Loss function	<code>torch.nn.functional.cross_entropy</code>
Mistral 7B weight datatype	bfloat16
Number of training epochs	300

Table B.1: Different functions and configurations used during training of J2A-2.x.

C | Prompt

```
<|im_start|> system
You are an expert AI assistant who is knowledgeable about music
↪ production, musical structure, music history, and music styles, and
↪ you are hearing audio of a short clip or loop extracted from a piece
↪ of music. IMPORTANT!! Do not repeat yourself or use redundant words.
↪ DO NOT use contradicting describing words, like "fast" and "slow",
↪ "light" and "dark", and "high pace" and "low
↪ pace".<|im_end|><|im_start|> user
Provide a detailed musical description of the clip, from the perspective
↪ of a musical expert describing the clip as they hear it being played.
↪ Make sure to describe the musical style, any unique features of the
↪ clip, its chords, tempo, and the instruments used.
↪ <|im_end|><|im_start|> assistant
```

Listing C.1: Prompt used for J2A-2.x

D | J2A-2.x Responses

Label	J2A-2.0 0.14	J2A-2.1 0.16
Uptempo pop rap music ideal for easy listening accompanied by rap male vocal, synth and drums.	1980's style special effects, groovy funky bass lines, and drum beat of this punk song. This song sounds like growing cells in a lab experimenting room. This song may be playing at a school-graduation	Reggae style vocal female singing with drums and bass lines. Sounds sex appeal. Ideal dance great rhythm for a sleepover. Great for mood lifting. Sounds 90's. Great bedroom karaoke. Great for a smoking shop

Table D.1: Random sample of label and inferences. The F1 score is written in the top right.

Label	J2A-2.0 0.22	J2A-2.1 0.14
A male singer sings this poignant melody. The tempo pop is slow with a steady drumming rhythm, cymbal crashes, guitar accompaniment, keyboard organ accompaniment and a strong bass line. The song is a poignant love song.	This recording is of an Allan Eaton song. Two male vocals are singing together while the male is singing in a high pitch. Synthesized pads are heard in the background. The recording is meant for playing in any fitness class	Calm hip hop featuring male rap with electronic elements and synth bass. Ideal for dancing in a funky yet casual environment. Read more into the groove. Perfect for dancing in a house party. Great running gait featuring female vocalist

Table D.2: Random sample of label and inferences. The F1 score is written in the top right.

Label	J2A-2.0 0.22	J2A-2.1 0.29
Energetic jiving pop music with peppy electronica. Music comprises of male lead vocal, synth, vocal harmonies and drums.	An unpolished male low gothic melodic voice of Western storyteller over deep reverb sounding seashores waves with drone, bass violin, percussion synthesized pad and guitar. If you want to med	a male baritone narration accompanied by male vocals and acoustic guitar with synth horns and ambient string notes leading into high paced electronic music part. Heard in a podcast. Interference is caused by Doppler effect.

Table D.3: Random sample of label and inferences. The F1 score is written in the top right.