UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Computer Science

Development of a Music Education Framework Using Large Language Models (LLMs)

Mudassar Amin

DTE–3900, Master thesis in Applied Computer Science, Narvik, May 2024

Abstract

This thesis explores the effectiveness of Large Language Models (LLMs) in enhancing educational methodologies, particularly focusing on personalized learning experiences in music education. Initially, a comprehensive literature review was conducted to establish the theoretical foundation and identify gaps in the current application of Large Language Models in education. Subsequently, employing a quantitative approach, the study utilized the Supervised Fine-Tuning QLoRA approach to adapt the Llama2-chat model to respond accurately to music educational queries. The Results showed the fine-tuned model with the instruction dataset provides some good results on the provided prompts. The performance of the model was evaluated using standard metrics such as BERTScore, F1 Score, and Exact_Match, which confirmed the model's efficacy in providing accurate and contextually appropriate responses. While the findings confirm the potential of integrating LLMs into educational frameworks, they also highlight some limitations, such as the need for continuous model training to adapt to evolving and diverse musical content and creativity. This study establishes a basis for future research, suggesting the exploration of symbolic music understanding models like MusicBERT and LLMs integration within music education.

Acknowledgements

I would like to express my deepest appreciation to all those who provided me the possibility to complete this thesis. A special gratitude I give to my supervisors **Bernt Arild Bremdal, Shayan Dadman and Kalyan R Ayyalasomayajula** for providing guidance, feedback on the work and recourse as well.

Furthermore, I am profoundly grateful to my parents and siblings, whose unwavering support have made it possible for me to pursue higher education. Their belief in my abilities and constant encouragement have been my strongest motivation throughout my studies.

Table of Contents

List of Figures

List of Tables

Abbreviations

AI Artificial Intelligence

LLM Large Language Model

PEFT Parameter-Efficient Fine-Tuning

QLoRA Quantized Low-Rank Adaptation

GPT Generative Pre-trained Transformer

BERT Bidirectional Encoder Representations from Transformers

NLP Natural Language Processing

VRAM Video Random Access Memory

RAG Retrieval-Augmented Generation

API Application Programming Interface

FP32 32-bit Floating Point

INT8 8-bit Integer

MIR Music Information Retrieval

RNN Recurrent Neural Network

LSTM Long Short-Term Memory

RBM Restricted Boltzmann Machine

VAE Variational Auto-Encoder

GAN Generative Adversarial Network

MIDI Musical Instrument Digital Interface

1 Introduction

In recent years, the capabilities of artificial intelligence (AI) have expanded significantly, particularly in the field of natural language processing (NLP). Large Language Models (LLMs), a sophisticated form of artificial intelligence, can generate and understand human language. This thesis explores the applications of LLMs in music education, specifically within music education. These models work by using a huge collection of written text to learn how to generate and understand human language [1].

Music education is challenged by various issues like limited access to quality instruction due to geographical and financial barriers, which lack students' ability to get high-quality music training. The diversity in students' learning styles and rates of progress complicates the effectiveness of traditional teaching methods. Traditional music education methods, which often depend on one-on-one lessons or small classes, are not easily scalable, limiting the availability of quality education.

This study [2] showed that Large Language Models (LLMs) enhance education by offering personalized learning experiences, serving as support tools. Additionally, LLMs provide assessments and feedback on student work and generate a variety of educational resources and content to enrich learning and teaching materials.

Large Language Models (LLMs) can automate tasks, but they also have some limitations such as biased output and hallucinations. To solve these issues for downstream tasks such as music education, it is necessary to control the model's response using fine-tuning method. Fine-tuning is the technique to increase the performance of LLMs in music education, where models are trained on music datasets in a supervised learning manner. This fine-tuning allows LLMs to understand and generate music content, therefore increasing their accuracy and performance in music education.

However, fine-tuning LLMs requires a lot of computational resources and VRAM due to

their larger size. To alleviate this problem, we can use parameter-efficient fine-tuning (PEFT) like Low-Rank Adaptation (LoRA) [3]. LoRA decreases the number of trainable parameters by introducing trainable low-rank matrices into the large model, significantly lowering the computational cost and memory resources for fine-tuning.

Despite LoRA's efficiency, the increasing size of Large Language Models (LLMs) still faces some challenges. To further optimize the process QLoRA [4] can be used which is the extended version of LoRA. QLoRA backpropagates gradients through frozen, 4-bit quantized pre-trained Large Language Models into Low-Rank Adapters (LoRA).

Using QloRA, we can fine-tune the Llama2 7B model with limited memory and resources. The Results showed the fine-tuned model with the instruction dataset provides some great results on the provided prompts. The performance of the model was evaluated using standard metrics such as BERTScore, F1 Score, and Exact-Match, which confirmed the model's efficacy in providing accurate and contextually appropriate responses. While the findings confirm the potential of integrating LLMs into educational frameworks, they also highlight some limitations, such as the need for continuous model training to adapt to evolving and diverse musical content and creativity. This study establishes a basis for future research, suggesting the exploration of symbolic music understanding models like MusicBERT and LLMs integration within music education.

1.1 Problem Statement

Traditional methods in music education often fail to meet the diverse needs and learning of students, they also face limitations such as the high cost of quality education. The development of Large Language Models (LLMs), a type of advanced artificial intelligence, offers an effective solution to these issues. However, using Large language Models effectively in music education needs a good understanding of how to use these LLM models to handle various musical-related questions. This thesis aims to develop a new framework that uses Large Language Models to make music education more personalized, engaging, and accessible for students and improve their creative and composition skills.

To explore the integration of LLMs in music education, this thesis is structured around several key objectives:

- **Literature Review:** Review the current state of research and identify potential gaps in the application of Large Language Models (LLMs) in music education.

- **Feasibility Study:** Conduct a feasibility study on using LLMs for interpreting and responding to musical inquiries, focusing on technical limitations and opportunities.

- **Framework Development:** Develop a music education framework that utilizes LLMs to provide feedback and ideas on simple music composition tasks, demonstrating LLMs' potential in targeted educational contexts.

- **Framework Evaluation:** Evaluate the effectiveness of the framework by analyzing

the quality of LLM-generated feedback on relevance and educational value.

To achieve these objectives, below mentioned research questions need to be answered, which we will discuss in the discussion section.

- **RQ1:** How can Large Language Models (LLMs) be effectively utilized to understand and respond to diverse musical queries and inputs in music education?
- **RQ2:** What are the main challenges in designing a framework for music education that provides personalized and adaptive learning experiences?
- **RQ3:** What are the key design principles when developing an AI-based music education framework?
- **RQ4:** How can user engagement and improvement in compositional skills be measured for such a framework?

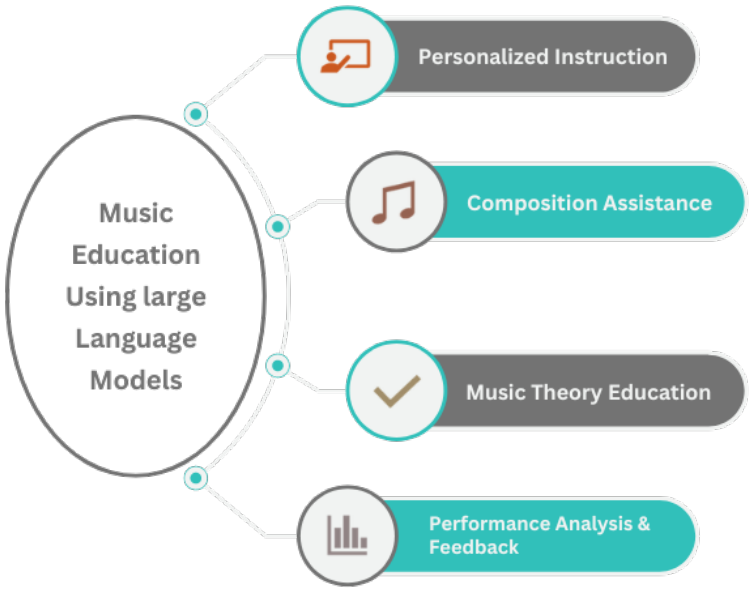


Figure 1.1: Applications of Large Language Models for Music Education

This thesis is structured into six chapters, Chapter 2 provides a comprehensive literature review, laying out the foundational theories and identifying research gaps. Chapter 3 outlines the tools, methodology, dataset preparation, and model selection and fine-tuning. Chapter 4 describes the training results and evaluates the model's performance, analyzing its effectiveness and efficiency. Chapter 5 discusses the findings, and research questions and explores the study's limitations. The thesis concludes with Chapter 6, which summarizes the research findings and suggests directions for future work, emphasizing the potential and challenges of integrating LLMs into music education frameworks.

2

Literature Review

This chapter explores artificial intelligence (AI) in music education, also focusing on the use of Large Language Models (LMMs) which are based on transformer architectures. This section explains how artificial intelligence tools are used to increase music education by showing the improvements in student performance for music education, especially for piano learning.

This chapter also briefly discusses transformer architectures which are widely used in Large Language Models. The chapter further discusses innovative systems such as MusiLingo [5], which integrate music encoders with Large Language Models to improve music captioning and query responses, thus increasing the educational experience. Moreover, different neural network architectures from simple melody generators to sophisticated systems capable of composing complex musical pieces are reviewed. Furthermore, we discussed some techniques for optimizing the outputs of Large Language Models (LMMs) for specific downstream tasks, to handle challenges such as bias and hallucination in model responses.

2.1 Artificial Intelligence in Music Education

As technology progresses it becomes increasingly vital to use modern technologies, especially artificial intelligence (AI) in education. AI has become an increasingly popular medium in a few years, which is being used by not only different companies but also the education sector like universities for education purposes. The study [6] shows the integration of artificial intelligence (AI) in music education, especially its uses in piano classes through AI-powered chatbots across seven music schools. The incorporation of AI into the educational process showed a 15% increase in students' academic performance. The research highlights the potential of chatbots in enhancing music education, offering a new pathway for development. AI's role in creating individualized learning paths and assessing student performance underscores its capability to transform music education [6]. The integration of AI in online teaching has improved the quality and efficiency of education by supporting personalized learning experiences and leveraging big data for Music Information Retrieval (MIR). AI in music education presents an optimistic future for the music industry, promising continuous improvement and widespread application after overcoming current limitations[7]. Table 2.1 shows improvement statistics by using AI in education.

Aspect of Music Education	Improvement
Overall Academic Performance	15%
Piano Playing Skills	6.51%
Solfeggio and Music Literature	4%
Vocal Singing	0.56%

Table 2.1: Improvements in music education through AI integration[7]

2.2 Large Language Models (LLMs)

Large Language Models use transformer models, are deep learning algorithms, and are trained using massive datasets that allow LMMs to predict, translate, or generate text and understand the natural language [8].

A transformer model, commonly used in Large Language Models, features an encoder and decoder to process data. It tokenizes input, breaks down input text into understandable pieces, and uses mathematical calculations to identify relationships between tokens. Unlike traditional long short-term memory models, transformer models employ self-attention mechanisms, allowing them to learn faster and consider entire sentence contexts for more accurate predictions [9].

2.2.1 Transformer Architecture

Transformer architecture introduced by [9] forms the foundation of modern Large Language Models. This architecture enhances text processing and generation capabilities by effectively capturing the relationships between words and phrases within their contexts. Below in Figure 2.1 is a transformer architecture with details of its key components.

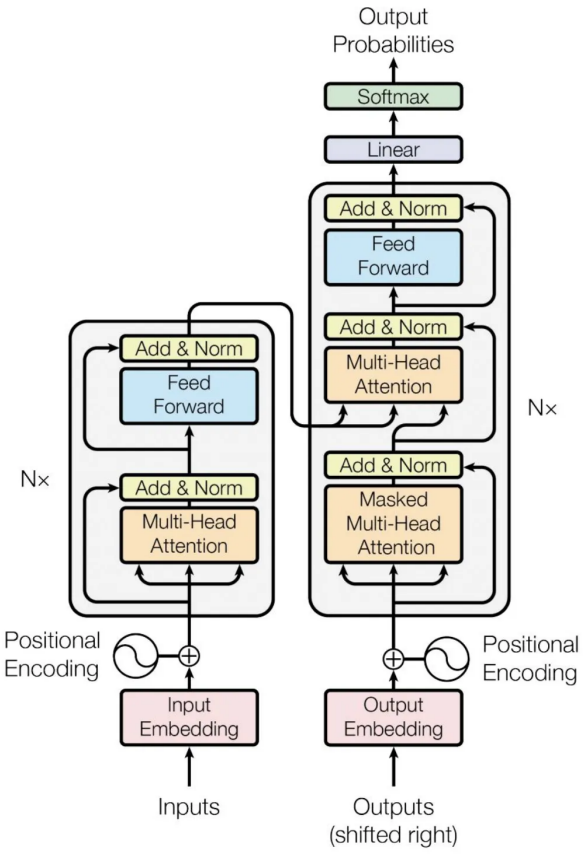


Figure 2.1: Transformer architecture introduced by [9] in 2017 paper "Attention is All You Need"

Self-Attention Mechanism The core feature of the transformer is the self-attention mechanism, which assesses the significance of all words in the sequence for each word within the sequence. It assigns weights to words based on their relevance in a given context, enabling a deeper and more nuanced understanding of language nuances.

Encoding and Decoding Layers Transformers are structured with multiple encoding and decoding layers. The encoding layers process the input text through a combination of self-attention and feedforward neural networks, developing a rich, contextualized representation of each word. Subsequently, the decoding layers focus on generating the output text, and predicting subsequent words by leveraging the context provided by the encoding layers.

Multi-Head Attention An extension of the self-attention mechanism, multi-head attention allows the transformer to simultaneously concentrate on various aspects of the input data. This feature enables the model to capture a broader range of relationships within the text, enhancing its interpretative and generative capabilities.

Feedforward Neural Networks Following the self-attention phase, transformers utilize feedforward neural networks to further refine the contextual representations of words. This step ensures that the nuances captured by the attention mechanisms are effectively integrated into the final model output.

There are different LMM architectures widely used for their unique capabilities and functionality, we are not going into the details of each but we have mentioned below a few of them.

- GPT (Generative Pre-trained Transformer)
- LLaMA (Large Language Model by Meta)
- BERT (Bidirectional Encoder Representations from Transformers)
- T5 (Text-to-Text Transfer Transformer)

In recent years "OpenAI has developed GPT-3, an advanced AI capable of generating human-like text. GPT-3, which stands out for its use of 175 billion parameters, was trained on Microsoft's Azure AI supercomputer at an estimated cost of $12 million. Since its beta release in June 2020, GPT-3 has showcased its ability to autonomously produce high-quality text" [10].

2.3 Music Representation

The selection of representation and its encoding is closely linked to how the input and output are configured within the architecture, specifically the number of input and output nodes. While a deep learning architecture has the capability to autonomously extract meaningful features from data, the chosen representation can greatly impact both the accuracy of the learning process and the quality of the output generated [11]. In this study, we discuss two primary representations: audio and symbolic.

2.3.1 Audio Representation:

Audio representation, particularly in the form of a signal waveform, provides a foundational digital model of sound. Signal waveform derived through Fourier transform, the primary benefit of using waveform is that it represent the foundational aspect of audio representation that is important for understanding and manipulating sound in its digital form. The main drawback is the computational demand processing low-level raw signals requires significant memory and computational resources.

2.3.2 Symbolic Representation:

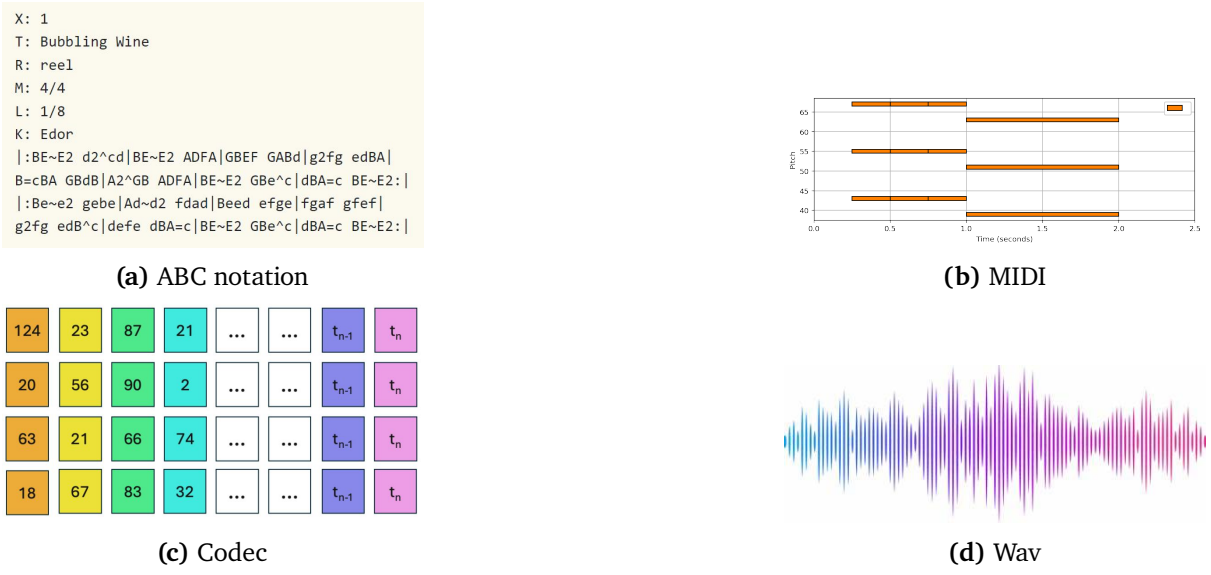
Symbolic representation in music involves extracting audio information into a format that allows easier manipulation and analysis of musical elements. Unlike raw audio data captured in a waveform, symbolic forms such as MIDI, Piano Rolls, and ABC Notation provide structured ways to represent musical notes, rhythms, and other information, Figure 2.2 shows symbolic formats. These formats help in tasks like composition and music generation and are also important in music information retrieval and computational musicology

where detailed manipulation of musical components is required.

- **MIDI:** Musical Instrument Digital Interface (MIDI), is a technical standard that outlines a protocol for events, a digital interface, and connectors, used in various electronic musical instruments, software, and devices. MIDI messages are crucial for indicating the start (Note on) and end (Note off) of a note. Each MIDI note number corresponds to a note pitch, ranging from 0 to 127 [11]. For symbolic music generation, some common musical events are listed in the Table 2.2 [12].

- **Piano Roll:** It is represented as a two-dimensional grid where the x-axis denotes successive time steps and the y-axis denotes pitch.

- **ABC Notation:** [13] Primarily used for folk and traditional music, this format encodes each note as a token. The pitch class of a note is denoted by a corresponding letter in English notation (e.g., A for A or La), with additional notations for octaves and duration. Measures are demarcated with bars ('|'). ABC notation offers a high compression rate, which results in shorter sequence lengths compared to MID and this format is also compatible with the language model for musical analysis and generation as explained in ChatMusician [14].



(a) ABC notation

(b) MIDI

(c) Codec

(d) Wav

Figure 2.2: Musical formats include Wav, Codec, MIDI (depicted as a piano roll), and ABC notation. As we move from figures (a to d), the compression rate decreases [14].

Event Type	Definition
Note On	Initiates a note at a specific pitch.
Pitch	Specifies the pitch value of a note.
Note Off	Signals the termination of a note at a specific pitch.
Note Duration	Calculated from the time difference between 'Note On' and the corresponding 'Note Off', including any 'Time Shift' events.
Time Shift	Advances the current time by a set number of quantized steps.
Position/Sub-beat	Identifies discrete positions within a bar.
Bar	Indicates the divisions between bars.
Piece Start	Denotes the commencement of a musical piece.
Chord	Represents each chord change.
Program/Instrument	Assigns a MIDI program number at the start of each track.
Track	Delineates each separate track within the composition.
Time Signature	Signifies changes in the time signature throughout the piece.
Note Velocity	Sets the intensity for subsequent 'Note On' events.
Tempo	Adjust for variations in the tempo throughout the piece.

Table 2.2: MIDI event types

2.4 Language Models for Music Captioning and Query Response

Large Language Models are being used for music captioning, recommendation, and query responses, MusiLingo [5] is an innovative system designed to bridge the gap between music and language, making it easier for people without formal musical training to interact with and understand music through natural language queries. It builds on previous models like MusCaps [15] and MuLan [16] by integrating pre-trained music encoders with Large Language Models (LMMs) for improved performance in music captioning and question-answering.

The system has been trained on the MusicInstruct Dataset, featuring over 60,000 Q&A pairs covering a wide range of music-related topics, demonstrating state-of-the-art performance in both objective and subjective music questions across various datasets. Despite its advancements, MusiLingo[5] has room for improvement in fine-tuning, model universality, and aligning generated responses with human music interpretations, presenting opportunities for further development[5]. Figure 2.3 shows the musiclingo model.

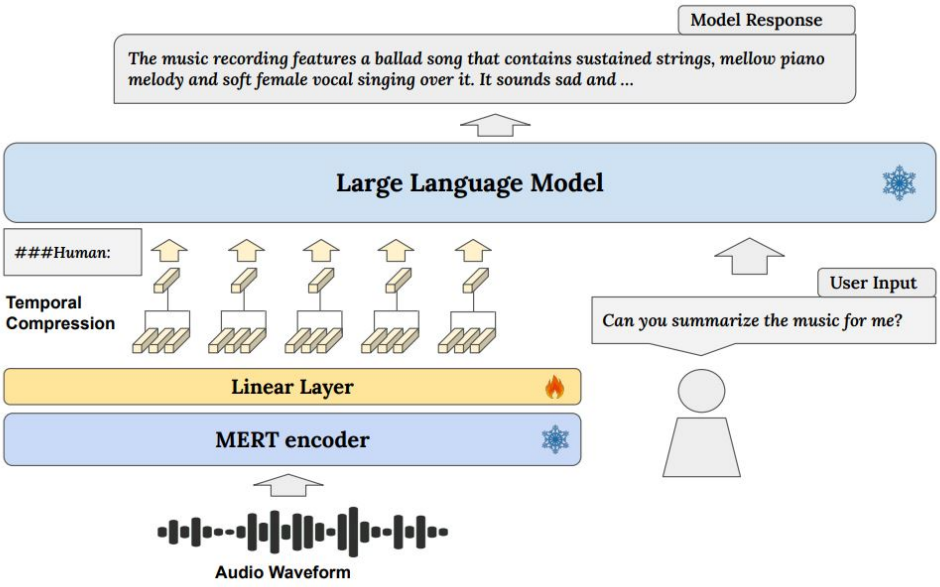


Figure 2.3: MusiLingo model overview [5].

Authors in Musilingo used MERT [17] as a music encoder to extract musical details from the input music clip. Additionally, they used Vicuna [18] as the large language model. It receives the music embeddings from the adaptation layer and produces text responses, incorporating additional user-provided text input.

2.5 Music Understanding and Generation

Recurrent Neural Networks (RNNs) were used for their proficiency in processing sequential data, marking their first significant application in 1989 with the generation of simple monophonic melodies. The Long Short-Term Memory (LSTM) networks later addressed the RNNs' limitations in preserving long-term sequence information, which facilitated their use in crafting structured blues improvisations [19].

Subsequent advancements brought about more sophisticated models such as the Recurrent Neural Network - Restricted Boltzmann Machine (RNN-RBM) [20], alongside improvements to RNNs that increased the generation capabilities for polyphonic music by better handling long-term structural dependencies. This narrative emerged with the introduction of advanced deep generative models like Variational Auto-Encoders (VAE), Generative Adversarial Networks (GAN), and Transformers. These cutting-edge models significantly expanded the frontiers of the field, enabling the creation of hierarchical models that capture extensive musical structures, GANs that produce melodies and complex multi-track compositions sequentially, and [21] used transformers to generate music with a longer-term structure as discussed in the paper [20].

[14] introduce an open-source transformer-based Large Language Model (LMM) "Chat-Musician" designed to unify music and language, enabling the model to understand and

generate music using ABC notation as a text-compatible representation. The ChatMusician-Base model was initialized using fp16-precision and based on the LLaMA2-7B-Base weights. It underwent a training process involving continual pre-training and fine-tuning, with LoRA adapters incorporated into attention and MLP layers, alongside comprehensive training across embeddings and all linear layers. To enable musical abilities in the LMMs they used musical dataset for its pretraining, musicpile was prepared using a wide range of sources, including general public datasets like Pile, Falcon RefinedWeb, and Wikipedia, further enriched with 255k music knowledge Q&A pairs and summaries generated using GPT-4. This work related to our task where we need to train the large language model so that it can understand the music and respond to those queries so that we can use the language model for music educational purposes or integrate it with a current educational platform on my making new framework. Empirical evaluations show that ChatMusician outperforms existing models like GPT-4 in music composition and understanding. Table 2.3 shows the performance of different models on music theory bench.

Models	**Music Knowledge**	**Music Reasoning**
Baseline	25.0	25.0
GPT-4	58.2	25.6
ChatMusician-Base	40.2	27.1
ChatMusician	39.5	26.3
GPT-3.5	31.2	25.3
LLaMA2-7B-Base	33.3	24.7

Table 2.3: Performance of various models on music knowledge and reasoning metrics [14].

However, they utilize the music ABC notation as it is simpler and more human-readable, making it good for folk music and simple melodies. In our model we need to use the MIDI file format due to its richness in information, MIDI files contain detailed information about each note and control message, such as pitch, velocity, duration, timing, onset, and offset makes it a robust choice for deep learning models and music composition [22]. ChatMusician mostly produces music in the Irish style, a bias that arises from an imbalance of this genre in the training data. This issue illustrates the model's difficulty in creating a diverse array of musical styles, pointing to a crucial area for future improvements. Moreover, the model occasionally generates 'hallucinations' which means outputs that are musically incorrect or irrelevant.

Authors in [23] discussed the generation of music using deep learning and all the different models how they used to generate the music. They also highlighted its rapid advancements and current challenges. They mentioned that while these models generate complex musical pieces. Yet its application faces significant challenges that impact the effectiveness and creativity of generated music for example they lack interaction, creativity, control, and structural coherence [24]. Furthermore, their study suggests exploring multi-agent systems and reinforcement learning to enhance creativity and adaptability, aiming for more dynamic and responsive music generation tools that better align with human musical processes.

2.6 MusicBERT for Symbolic Music Understanding

MusicBERT [25] is a large pre-trained model specifically designed for understanding symbolic music data. Symbolic music data (MIDI) represent music in a structured format that includes details like pitch, duration, and velocity of notes. This model is trained on a wide range of datasets containing over a million songs, allowing it to understand a large variety of musical styles and structures. As mentioned in the paper [25] MusicBERT model is very effective for the four main tasks related to music understanding, melody completion, accompaniment suggestion, genre classification, and style classification.

MusicBERT includes features discussed in the paper[25] like OctupleMIDI Encoding and Bar-level Masking Strategy that improve its performance. OctupleMIDI Encoding combines several music details into one piece, which helps simplify and speed up how the model learns from complex music in other words. They use the sequences of octuple tokens to represent symbolic music (MIDI) and OctupleMIDI reduces the sequence for example it will encode 6 notes into 6 tokens which is far shorter than the other encoding methods like CP-Like encoding and REMI-Like encoding. The Bar-level Masking Strategy helps the model learn more effectively by covering up entire bars of music, which prevents the model from just copying nearby notes and encourages it to understand deeper patterns.

By using MusicBERT's large Million MIDI Dataset, which contains a wide variety of music, our model can get better at recognizing and creating complex musical pieces that help students make melodies composition and analyze the student MIDI files. Adding these features will help make our model more versatile and creative, potentially improving how it's used in music education and setting a new Framework for the use of LLMs in Music Education.

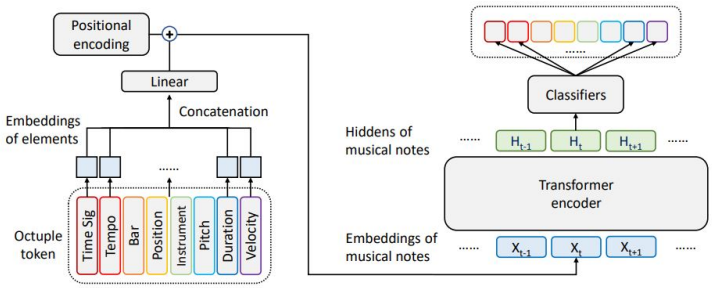


Figure 2.4: MusicBERT model structure [25].

2.7 Techniques for Enhancing LLMs Performance for Downstream Tasks

Large Language Models (LLMs) are trained on vast amounts of data across multiple domains, and they have some limitations such as bias in the generated response and hallucination, which means the context is misunderstood and sometimes generates incorrect and false text. However, when targeting specific tasks, they require fine-tuning with

task-specific datasets to optimize performance. Here we will discuss some techniques as shown in Figure 2.5 that can be used to control the model response.

2.7.1 Prompt Engineering

Prompt Engineering means carefully crafting prompts that can effectively guide the model's output. While prompting, we can use specific keywords and instructions to get the desired output. In a few shots, learning prompts include some examples to guide the model's output, this can include setting the tone, style, or even the format of the output. Prompting is an iterative approach that is time-consuming though it's a powerful technique still faces challenges such as ensuring consistency and the need to update the prompts for different cases.

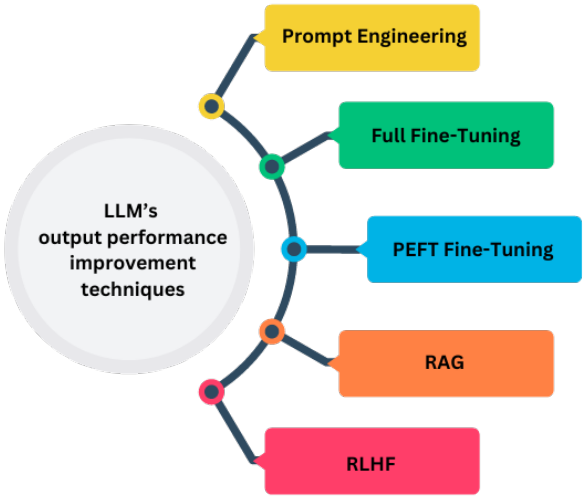


Figure 2.5: LMM's output performance improvement techniques

2.7.2 Fine-Tuning

Fine-Tuning process entails adapting a pre-trained model by training it on a smaller, task-specific dataset. Full fine-tuning modifies the all model's weights to align with the requirements of the downstream task, enhancing the accuracy and relevance of its outputs. Comprehensive fine-tuning is an efficient approach as it leverages a pre-trained model's existing general language knowledge, requiring less data to adapt to specific tasks. This technique notably enhances accuracy by refining the model's ability to capture and reproduce the nuances of specialized domains, crucial in fields laden with unique terminologies such as the legal, medical, or financial sectors. Additionally, by training on a diverse array of examples, including edge cases, the model becomes more adept at handling a wide range of inputs, thereby increasing its robustness and overall performance. Effective fine-tuning of these Large Language Models often requires advanced computational resources with large memory usage. Furthermore, the full fine-tuning method is not only time-consuming but also requires the distribution of computational tasks across multiple GPUs [26] [27].

2.7.3 Parameter Efficient Fine-tuning (PEFT)

Parameter-efficient fine-tuning increases the efficiency of learning capabilities for new tasks by updating a small number of the pre-trained model's parameters. PEFT [28] handles the issues of computational cost and memory usage for the full fine-tuning of Large Language Models. This approach involves maintaining most of the pre-trained model's parameters frozen while only changing those needed for specific tasks. Parameter-efficient fine-tuning has different techniques but the most used for these cases are Adapter, LoRA, and QLoRA [29].

LoRA: Low-Rank Adaptation (LoRA) [3], is an efficient approach for fine-tuning Large Language Models. LoRA reduces the number of trainable parameters and makes the process feasible for fine-tuning large models. LoRA works by adding trainable low-rank components into each layer of a transformer model while keeping the pre-trained model weights frozen. This way we can reduce the computational cost for training and the memory requirements for GPU resources.

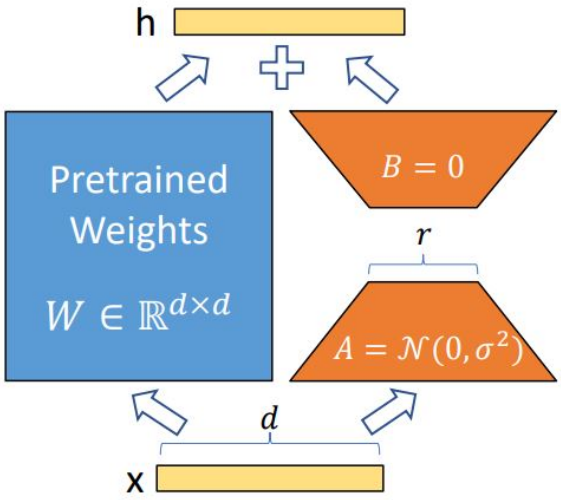


Figure 2.6: LoRA reparameterization trains only A and B [3].

Suppose we have a LLM with a weight matrix dimension of 50,000 x 50,000. In a traditional full fine-tuning scenario, we would need to update 2,500,000,000 parameters. However, by employing LoRA, we can significantly reduce this number.

Using LoRA, we decompose the large weight matrix into two smaller matrices A and B, each with a low rank r. For instance, if we choose $r = 3$, the new parameter count becomes $(50,000 \times 3) + (3 \times 50,000) = 300,000$ parameters. This represents a reduction by a factor of more than 8,000, dramatically decreasing the computational load and memory footprint needed for training.

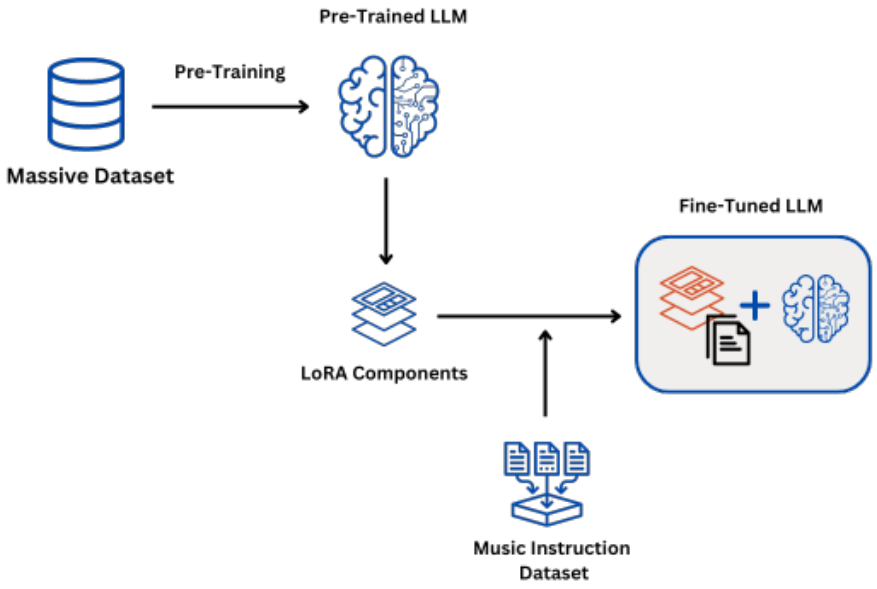


Figure 2.7: LoRA parameter-efficient fine-tuning process

If LoRA is good then why do we need QLoRA? First, we will discuss the concept of quantization. Quantization is a technique used to reduce the precision of model weights from a continuous set of values to a discrete one, thereby shrinking the model's memory footprint and decreasing computational demands during inference. It involves converting weights from floating point to integer values by scaling according to a normalization constant and then rounding off. While quantization typically leads to reduced storage requirements and energy consumption, it can sometimes increase inference time due to the overhead of performing these operations. However, with advanced methods like Quantization Aware Training (QAT) [30], models can be fine-tuned to incorporate quantization errors, allowing them to perform comparably to their non-quantized versions. This makes quantization especially valuable for deploying deep learning models on edge devices with limited computational resources [26].

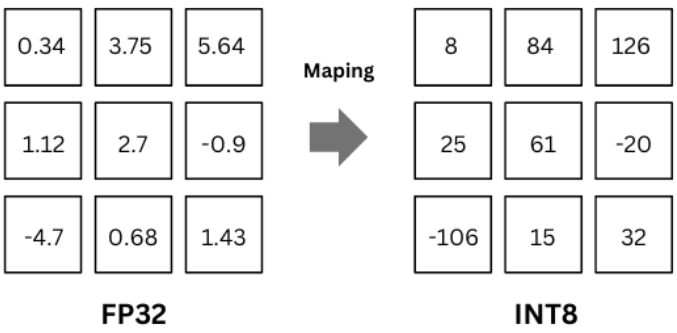


Figure 2.8: Quantization from FP32 to INT8

QLoRA: QLoRA [4], a cutting-edge fine-tuning approach that dramatically reduces memory usage, enabling the fine-tuning of a 65 billion parameter model on a single 48GB GPU while retaining full 16-bit fine-tuning task performance [4]. This breakthrough is facilitated by QLORA's ability to backpropagate gradients through a frozen, 4-bit quantized pre-trained language model into Low-Rank Adapters (LoRA). This Study describes their most effective model family, termed Guanaco, which exceeds all previously released models on the Vicuna benchmark, achieving 99.3% of the performance of ChatGPT with just 24 hours of fine-tuning on a single GPU.

QLORA incorporates several key innovations to minimize memory usage without sacrificing performance:

4-bit NormalFloat (NF4) The first innovation is the 4-bit NormalFloat (NF4), a novel data type optimized for normally distributed weights, enhancing efficiency in memory use. The primary quantization equation used in QLORA is:

$$X_{\text{int8}} = \text{round}\left(\frac{127}{\text{absmax}(X_{\text{FP32}})} \times X_{\text{FP32}}\right) \tag{2.1}$$

where X_{FP32} is the tensor of model weights in full precision, and X_{int8} is the quantized tensor in 8-bit format. This normalization and scaling process compresses the data without significant loss of information.

Double Quantization The second innovation is Double Quantization, which involves quantizing the quantization constants themselves, significantly reducing the average memory footprint. The dequantization process critical for restoring the quantized values to their original scale during model inference is described by:

$$\text{dequant}(c_{\text{FP32}}, X_{\text{int8}}) = X_{\text{int8}} \times c_{\text{FP32}} = X_{\text{FP32}} \tag{2.2}$$

This ensures that performance remains unaffected even at reduced precision.

2.7.4 Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) [31], developed by Meta researchers, revolutionizes natural language processing by integrating information retrieval with text generation. RAG enhances Large Language Models (LMMs) by fetching relevant contextual data from a vast vector database, improving the responses' precision and relevance. This is achieved by embedding data into vectors, retrieving pertinent information based on user queries, and concatenating these findings with the original query to enrich the model's context.

RAG is invaluable in applications requiring up-to-date or extensive document-based knowledge, such as technical support systems utilizing internal manuals or medical research databases. Unlike simple prompt engineering that struggles with limited context, RAG broadens the LMM's understanding by incorporating real-time, specific information, thereby minimizing inaccuracies and 'hallucinations'—incorrect assumptions made by the model [32].

The advantages of using RAG include its ability to update responses with evolving data,

traceability of information sources, and cost-effectiveness, as it reduces the need for extensive labeled data and computational resources. However, RAG has some limitations like it may not substantially improve outcomes in scenarios where LMMs fail to process complex data like financial summaries or medical records. In such instances, fine-tuning the LMM might be more effective than augmenting it with RAG, particularly when a deeper understanding of specialized content is required [27].

These techniques are designed to optimize Large Language Models for specific tasks, resolving limitations like bias, hallucinations, and computational inefficiencies. Prompt engineering and fine-tuning provide foundational performance improvements, while PEFT and RAG offer advanced solutions to enhance the performance of Large Language Models. This thesis will integrate the PEFT method to create a more accurate, efficient, and context-aware model.

3

Methods

This chapter discusses some important steps taken for this thesis including which tools we used, dataset preparation, and the main methodology for fine-tuning the Llama2 7billion parameters model for our music education framework. Moreover, it also discusses how we effectively utilized the available T4 GPU within the limited VRAM of Google-Colab using the QLoRA technique. The methodology flowchart Fig 3.1 outlines our important steps for the model fine-tuning process. Moreover, the Chapter discusses how we created the dataset and why we used the parameter-efficient fine-tuning QLoRA approach. The code for this thesis is available on the git repository [33].

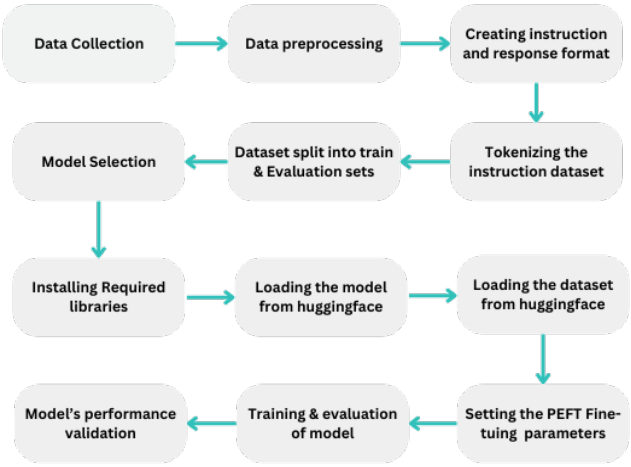


Figure 3.1: Methodology flowchart for this study

3.1 Tools

We used the pre-trained Llama2 chat model, which has 7 billion parameters. The fine-tuning process was carried out using a T4 GPU of Google-Colab, chosen for its long availability and cost-effectiveness despite its limited 16 GB VRAM. This memory size is not enough for the full fine-tuning. To address this issue, we implemented the parameter-efficient fine-tuning method, QLoRA, which supports 4-bit precision to optimize VRAM usage effectively. Our implementation made extensive use of Huggingface's libraries, including transformers, accelerate, peft, trl, and bitsandbytes, to facilitate the QLoRA technique.

Additionally, the development and data processing phases were conducted on a local environment using Jupyter Notebook on a laptop equipped with an Intel(R) Core(TM) i7-8550U CPU at 1.80GHz, with 1992 MHz, 4 cores, and 8 logical processors. Despite the laptop's limited RAM, this setup proved capable of efficiently coding, experimenting, and managing project files, thereby demonstrating the feasibility of conducting high-level research with accessible consumer hardware. Table 3.1 highlights some tools and technologies used for this thesis.

Tool/Technology	**Description**
Llama2 Chat Model	A pre-trained model with 7 billion parameters.
T4 GPU of Google-Colab	Used for the fine-tuning process with 16GB VRAM.
QLoRA	For 4-bit precision to effectively optimize VRAM usage.
Huggingface Libraries	Includes transformers, accelerate, peft, trl, and bitsandbytes to facilitate the QLoRA technique.
Jupyter Notebook	Software environment used for data processing on a local laptop.
Intel Core i7-8550U CPU	1.80GHz with 1992 MHz, 4 cores, and 8 logical processors, utilized in local environment setup.
Python	The programming language used throughout the project for scripting and automation tasks.
music21 Library	A Python library used for midi analysis and manipulation.

Table 3.1: Summary of tools and technologies used in the project

3.2 Instruction Dataset

Instruction datasets in the form of Q&A pairs are used to fine-tune the Large Language Models LLMs. Fine-tuning is usually undergoes a supervised machine learning approach and contains both strings input as instruction and output as a response. There are many instruction datasets with different formats and lengths some of them are created manually like the Flan Collection[34] and Dolly15k[35] dataset while others are created using Large Language Models like the Alpaca[36].

For a music education purpose, we have created a small instruction dataset as the purpose is to finetune the LLM on this dataset so that the model can understand diverse instructions and responses to them related to music education without Hallucination. In AI and language

models, "hallucination" describes situations where these models produce information that is incorrect, made up, or nonsensical as though it were factual. Such errors can arise from a variety of factors, such as the data used to train the model, how the model processes input, or its limited grasp of the context[37].

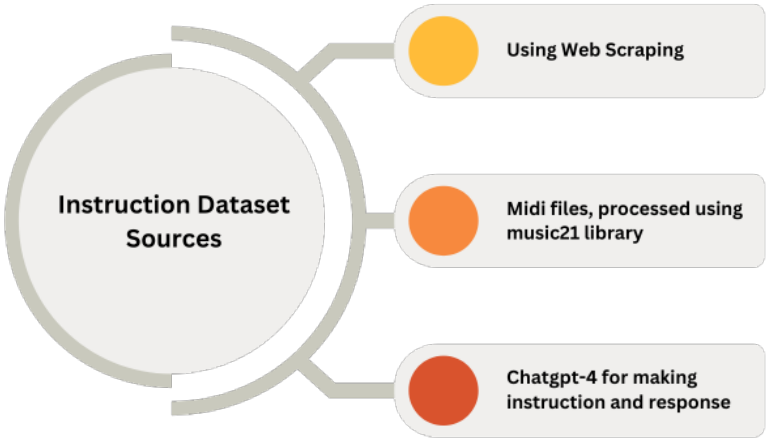


Figure 3.2: Sources used to prepare instruction dataset

We used MIDI files, web data, and ChatGPT-4 to create a rich and diverse instruction dataset. Initially, we extracted the notes and chords from the midi files utilizing the music21 Python library, a powerful tool for analyzing and manipulating musical data. Following this extraction, we provided the chords and notes to ChatGPT-4, which then generated detailed instruction and response pairs. The Instruction dataset was saved in JSON format, to easily accessible data for further processing and analysis with LLMs. As we used the Llama2 7B parameters model, the chat format is recommended as follows:

```
<s>[INST] <<SYS>>{System Prompt}</<SYS>>
{Instruction}[/INST]
{Response}</s>
```

For the Large Language Models to easily process and understand human language, we converted the dataset into chunks or *tokens*. Tokens are fundamental step in preparing data for these models. Tokenization of data is the process of breaking down text into smaller, manageable chunks, known as tokens, which simplifies the computational understanding and analysis of the language. This procedure is vital as it increases the efficiency of the large language model during training and fine-tuning by providing a structured form of data that mimics the natural language patterns. we uploaded the tokenized dataset to Hugging Face, ensuring it is readily available for future use and can be easily accessed by the global research community. Fig 3.3, Fig 3.4, and Fig 3.5 show the distribution of tokens on instruction, response, and chat columns respectively. Figure 3.6 shows the final instruction dataset that is used for this thesis, the "instruction" columns contain all the instructions the "response" column contains all the responses, and the last column which is "chat", includes both instruction and response in the format above mention.

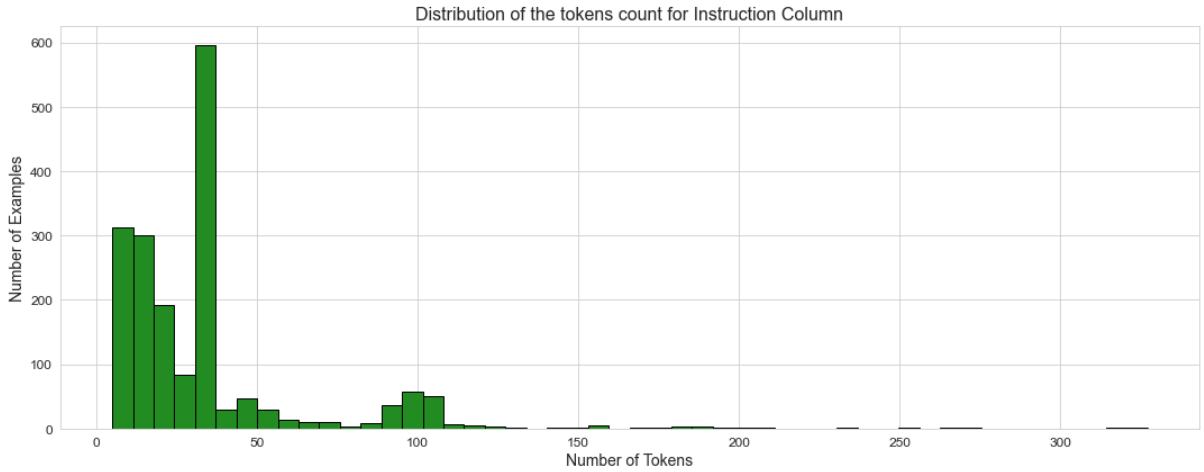


Figure 3.3: Distribution of tokens on instruction column

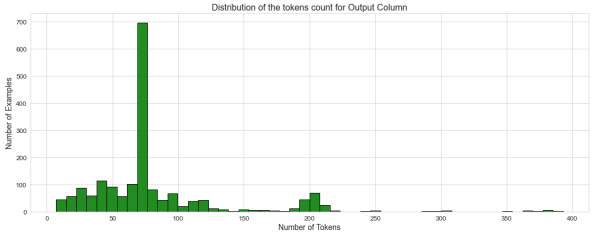


Figure 3.4: Tokens distribution on response column

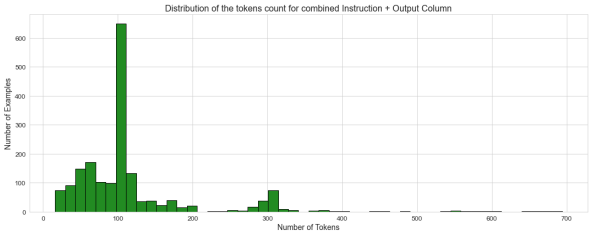


Figure 3.5: Tokens distribution on chat column.

response string · lengths 28 842	instruction string · lengths 12 964	chat string · lengths 73 1.6k
A C major chord is constructed with a root (C), a major third (E), and a perfect…	What is a C major chord?	### Instruction: What is a C major chord? ### Response: A C major chord is…
The C major chord consists of three notes: C, E, and G.	How many notes make up the C major chord?	### Instruction: How many notes make up the C major chord? ### Response: The C…
For the left hand, the fingerings are little finger (5), middle finger (3), and…	What are the fingerings for playing a C major chord on the piano with the left…	### Instruction: What are the fingerings for playing a C major chord on the piano…
For the right hand, the fingerings are thumb (1), middle finger (3), and little…	What are the fingerings for playing a C major chord on the piano with the right…	### Instruction: What are the fingerings for playing a C major chord on the piano…
C/E is the first inversion of the C major chord, where E is the bass note. This…	What does C/E signify in chord inversions, and which inversion is it?	### Instruction: What does C/E signify in chord inversions, and which inversion is…
C/G is the second inversion of the C major chord, where G is the bass note. This…	What does C/G signify in chord inversions, and which inversion is it?	### Instruction: What does C/G signify in chord inversions, and which inversion is…

‹ Previous 1 2 3 ... 19 Next ›

Figure 3.6: Instruction dataset tokenized and pushed to the huggingface

3.3 Model Selection

In this section, we delve into the reasons behind our model selection and the array of alternatives that were considered. Subsequent sections will provide a detailed, step-by-step explanation of the fine-tuning process we employed.

A variety of models are currently available for tasks like ours, including high-profile options such as ChatGPT and PalM. These models, however, often come with associated costs for access to their APIs or for fine-tuning capabilities, as detailed in the pricing policies of their providers [38]. After evaluating the options, we decided to use the Llama 2-Chat model [39]. This model, with configurations ranging from 7 billion to 70 billion parameters, has demonstrated superior performance over other existing open-source chat models across various benchmarks. Specifically, the Llama 2-Chat is a fine-tuned version of the original Llama 2, optimized particularly for conversational applications, making it ideal for our requirements in educational technology.

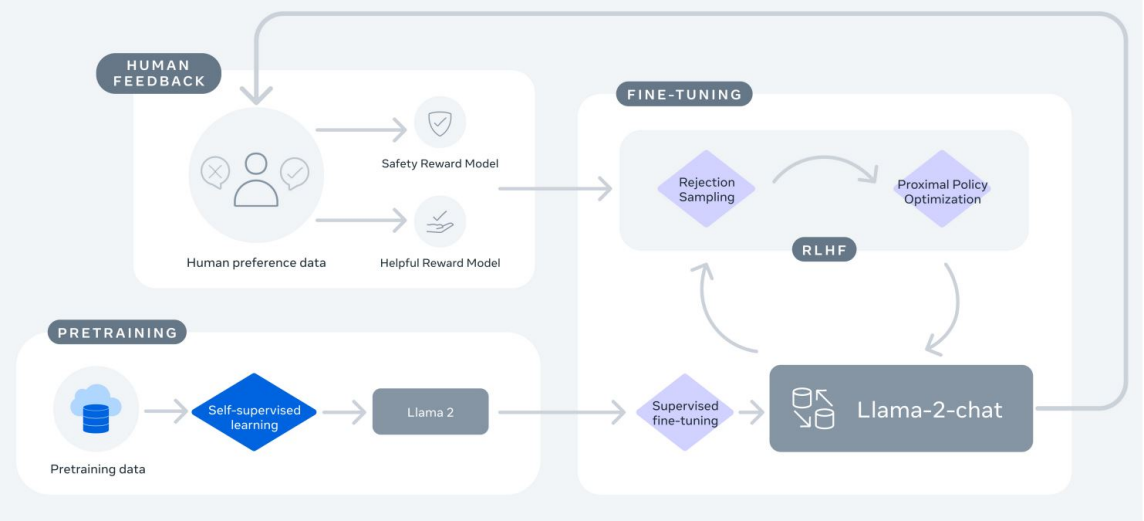


Figure 3.7: Training process of Llama 2-Chat provided by meta [39]

The fine-tuning environment is prepared by installing the required libraries from hugging face using pip install transformers, accelerate, peft, trl, and bitsandbytes. We used the Google Colab T4 16GB of VRAM GPU which is not completely enough to save the model's weight, so here we used the QLoRA technique (reviewed in Chapter 2) to fine-tune the model in 4-bit precision and optimize VRAM usage.

3.4 Quantized Low-Rank Adaptation (QLoRA)

Fine-tuning is the process of increasing the performance of Large Language Models LLMs or changing the behavior for the specific use case, with the increasing size of pre-trained Large Language Models the traditional fine-tuning process is impractical and the process is highly resource-intensive, with the fine-tuning of models like the LLaMA 65B requiring upwards of 780 GB of GPU memory as discussed in the papers[4][40] high computational cost has limited the fine-tuning of Large Language Models, so to handle this issue they proposed the new approach named QLoRA[4] which reduced the memory VRAM usage to fine-tune Large Language Models on a single GPU while keeping the 16-bit fine-tuning performance.

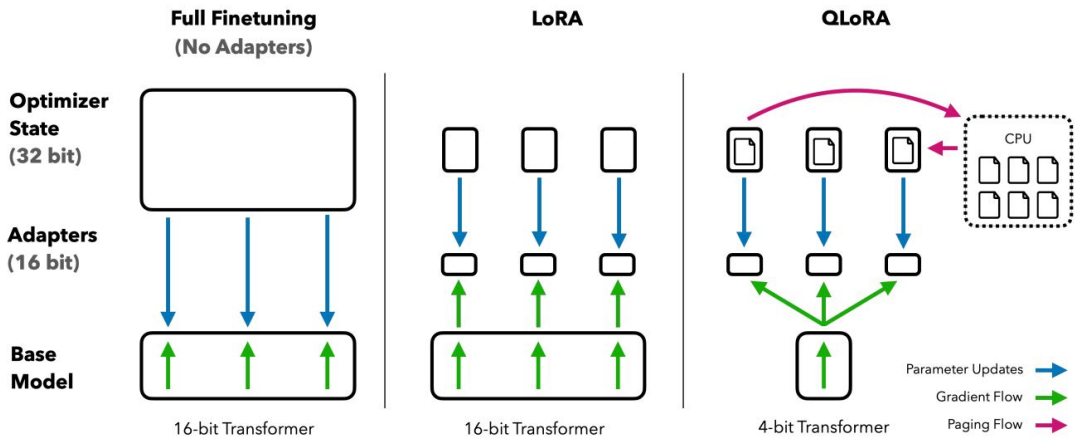


Figure 3.8: Image from the QLoRA paper showing the different methods fullfine-tuning, lora and QLoRA[4].

Three approaches (4-bit NormalFloat, Double Quantization and Paged Optimizer) introduced by QLoRA [4] that are intended to make fine-tuning language models more memory-efficient without affecting the performance. So for the Llama2 model with 7 billion parameters on a Google-Colab T4 GPU where we have 16 GB of VRAM, we unitized the 4-bit precision to optimize the VRAM usage, which means the model is quantized to 4-bit and then finetuned with LoRA for this purpose we used bitsandbytes, a quantization library with a Transformers integration. Using this we can quantize a model to 8 or 4 bits and allow many other options by configuring the BitsAndBytesConfig[41].

```
from transformers import BitsAndBytesConfig
quant_config = BitsAndBytesConfig(
    load_in_4bit=True,
    load_in_8bit_fp32_cpu_offload=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=compute_dtype,
    bnb_4bit_use_double_quant=False)
```

Authors in [41] state that setting `load_in_4bit=True` enables the quantization of the model to 4 bits, and `bnb_4bit_quant_type="nf4"` uses a specialized 4-bit data type for weights that are initialized from a normal distribution."

LoRA fine-tuning adds a few adaptable elements to models, enhancing their function while keeping memory usage low. It integrates an additional projection into the model's operations, increasing its efficiency without changing the original parameters. This method boosts performance and allows adding multiple adapters with little impact on memory [3][4][42]. We used the following value for LoRA configuration.

```
peft_params = LoraConfig(
    lora_alpha=16,
    lora_dropout=0.1,
    r=8,
    bias="none",
    task_type="CAUSAL_LM")
```

Now we will set the training parameters to optimize the training process.

```
from transformers import TrainingArguments

training_params = TrainingArguments(
    output_dir="./training_Results",
    evaluation_strategy="epoch",
    num_train_epochs=2,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    gradient_accumulation_steps=2,
    optim="paged_adamw_32bit",
    save_steps=25,
    logging_steps=25,
    learning_rate=2e-4,
    weight_decay=0.001,
    fp16=False,
    bf16=False,
    max_grad_norm=0.3,
    max_steps=-1,
    warmup_ratio=0.03,
    group_by_length=True,
    lr_scheduler_type="constant",
    report_to="wandb"
)
```

We passed all these components, such as the model, dataset, LoRa configuration, tokenizer, and training parameters, to the SFT Trainer. This is a key step in reinforcement learning. We use supervised fine-tuning, and the transformer reinforcement learning (TRL) provides an API for SFT models, which allow us to fine-tune Large Language Models on our own dataset hosted on Hugging Face with just a few lines of code. There are differences between the Trainer and SFTTrainer, both of which are Hugging Face classes, but they have different use cases. The Trainer is suitable for larger datasets and offers more customization of the training loop, while the SFTTrainer is more efficient for fast fine-tuning on small datasets with pre-trained models[43].

Feature	**Trainer**	**SFTTrainer**
Purpose	General-purpose training from scratch	Supervised fine-tuning of pre-trained models
Customization	Highly customizable	Simpler interface with fewer options
Training workflow	Handles complex workflows	Streamlined workflow
Data requirements	Larger datasets	Smaller datasets
Memory usage	Higher	Lower with PEFT and packing optimizations
Training speed	Slower	Faster with smaller datasets and shorter times

Table 3.2: Comparison of Trainer and SFTTrainer features[43]

3.5 Fine-Tuning Details

In our fine-tuning process, we started by setting the learning rate to 2×10^{-4} and used a weight decay of 0.001 to handle the overfitting issue . The main purpose is that our model training remains stable and efficient. The model processes information in batches of 4, each with a sequence length of up to 4096 tokens. This batch and sequence configuration is particularly effective in managing memory resources during fine-tuning and training of Large Language Models. Overall fine-tuning of our model used two epochs, this duration chosen to increase the performance without excessive training of model, which could lead to over-fitting or unnecessary computational cost. This balance is crucial for achieving high performance while maintaining efficiency. It is suggested that 3-5 epochs are enough for fine-tuning the model on small dataset like in our case where we have around 1800 rows of instruction.

After fine-tuning, it is important to save not just the model weights but also the configuration and tokenizer to ensure compatibility during future loads. The updated model pushed to the huggingface Llama2-Finetuned, provides an easy interface for interaction through the transformers pipelines listed below, which are ideal for testing and text-generation.

```
# Use a pipeline as a high-level helper
from transformers import pipeline

pipe = pipeline("text-generation", model="mudassar93/llama2-chat-piano")
```

```
# Load model directly
from transformers import AutoTokenizer, AutoModelForCausalLM

tokenizer = AutoTokenizer.from_pretrained("mudassar93/llama2-chat-piano")
model = AutoModelForCausalLM.from_pretrained("mudassar93/llama2-chat-piano")
```

In this chapter, we explored the instruction fine-tuning process of LLMs with basic computer resources. Our use of the T4 GPU from Google-Colab, despite its limitations, highlighted the necessity for optimization techniques like QLoRA to handle Large Language Models (LLMs) efficiently. The successful implementation of QLoRA, utilizing 4-bit precision to reduce VRAM usage, showcases an innovative approach to overcoming hardware constraints while maintaining model performance. Additionally, using Huggingface's powerful libraries facilitated the effective adaptation and execution of complex tasks, highlighting the potential of open-source tools in expanding the capabilities of AI research within existing technological limits.

4

Results

This chapter discusses the training loss, evaluation loss, and learning rate of our pre-trained model ensuring the fine-tuning process, and highlights the model's effective learning over each epoch. It also discusses the response generated by our models by providing some instructions where the models showed consistence performance. Moreover, the chapter explained some standard matrices to validate the model-generated text including BERTScore, F1, and Exact-Match. These metrics determine the model's precision, recall, and overall accuracy in generating text, showing how well they align with human evaluative standards, and showing its applications in natural language processing tasks.

4.1 Training & Evaluation

The training and Evaluation process shows a clear improvement in the loss over the epochs, which is a good thing that the model is learning effectively from our instruction dataset. Below we mentioned the training and evaluation loss:

Epochs and Losses

- **Epoch 0:** The training loss started at 0.4916 and the evaluation loss was 0.3797 This initial loss is relatively high, as expected at the beginning of training, but it indicates that the model has started to learn from the training data.

- **Epoch 1:** There was a significant decrease in both training and validation loss to 0.3104 and 0.3348, respectively. This decrease indicates that the model is effectively adjusting its weights and biases to minimize the error in its performance or generation task.

Evaluation

- **Validation Loss:** 0.33476176857948303, which relates to the loss reported at the end of Epoch 1, indicating consistent model evaluation.
- **Evaluation Runtime:** 26.0132 seconds, which provides a measure of how long it takes to evaluate the model on the validation dataset.
- **Evaluation Samples per Second:** 14.031. This metric shows how many samples are being processed per second during validation, providing insight into the model's inference speed.
- **Evaluation Steps per Second:** 3.537. This reflects the number of batches (steps) processed per second, offering an indication of the processing efficiency during model evaluation.

Epoch	Training Loss	Evaluation Loss
0	0.4916	0.3797
1	0.3104	0.3348

Table 4.1: Training and Evaluation Loss over epochs

Metric	Value
Validation Loss	0.3348
Evaluation Runtime (seconds)	26.0132
Evaluation Samples per Second	14.031
Evaluation Steps per Second	3.537

Table 4.2: Evaluation Metrics

Moreover, a dropout of 0.1 is used which helps to prevent model overfitting by randomly dropping units in the neural network during training, and for the optimizer, we used Paged AdamW (32-bit precision).

Figure 4.1: Shows the training loss over time during the fine-tuning of model

4.2 Generated Text Validation

We set the system prompt as *'You are a helpful assistant that provides accurate and concise responses related to music theory'*. The temperature was set to 0.1 and the output token size to 200, where the temperature controls the randomness of the output generation; lower temperature results in more predictable and conservative text, while the token size limits the length of the generated text. Here are some prompts given to the model and their responses.

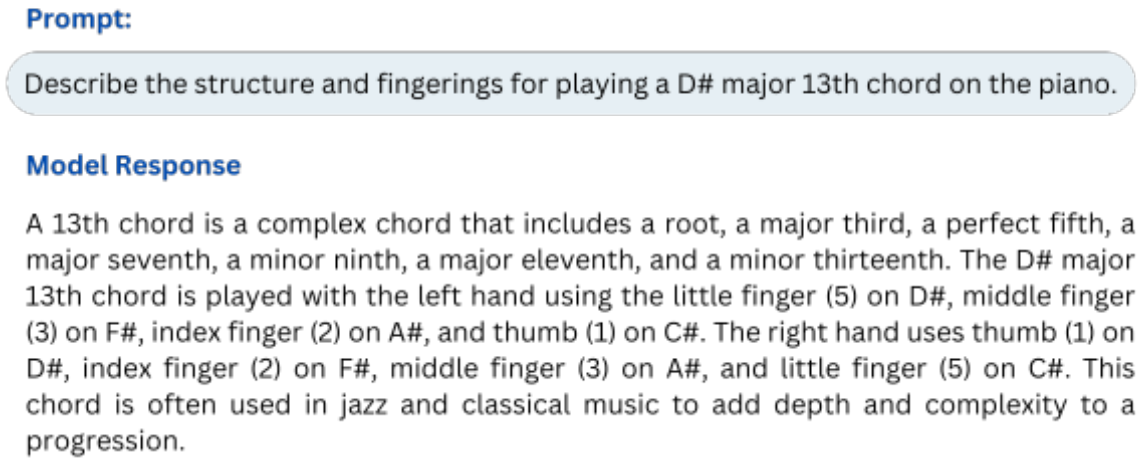


Figure 4.2: Prompt given and the response generated by model

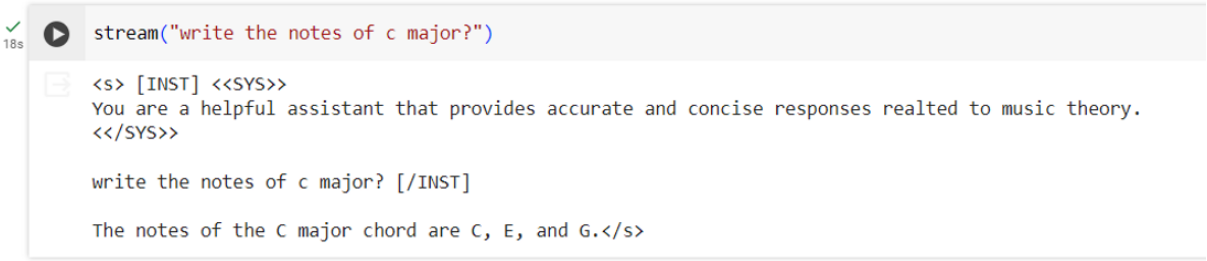


Figure 4.3: User prompt given to model "write the notes of C major".

```
stream("How to play D minor and how many notes are in D minor?")

<s> [INST] <<SYS>>
You are a helpful assistant that provides accurate and concise responses realted to music theory.
<</SYS>>

How to play D minor and how many notes are in D minor? [/INST]

To play the D minor chord on the piano, use the fingerings: Left hand - little finger (5) on D, middle finger (3) on F, thumb (1) on A;
Right hand - thumb (1) on D, middle finger (3) on F, little finger (5) on A. This chord consists of three notes: D, F, and A.
The D minor chord is constructed with a root (D), a minor third (F), and a perfect fifth (A). It is abbreviated as Dm.</s>
```

Figure 4.4: User prompt given to model "How to play D minor and how many notes are in D minor".

Model evaluation is an important step that involves measuring the performance, precision, and effectiveness of Large Language Models across different language-related tasks [44]. Our fine-tuned model's generated text is validated using the standard matrices BERTScore, F1, and Exact_match these are metrics used to evaluate the performance of Large Language Models, particularly in tasks involving natural language processing (NLP) and machine learning. Each of these standard metrics provides a distinct purpose and offers insights into different aspects of model performance[45].

4.2.1 BERTScore

BERTScore is a standard metric designed to evaluate text generation more effectively than traditional methods. Unlike conventional metrics that focus on exact token matches between a generated sentence and a reference, BERTScore leverages contextual embeddings to assess the similarity between each token in the candidate and reference sentences. The results demonstrate that BERTScore not only aligns more closely with human evaluations but also surpasses existing metrics in guiding model selection. Additionally, in an adversarial paraphrase detection scenario, BERTScore proved to be significantly more resilient, effectively handling complex examples where traditional metrics failed[46].

Metric	Value
BERTScore F1	0.94009
BERTScore Precision	0.91825
BERTScore Recall	0.96299

Table 4.3: Key performance metrics of our model

- **BERTScore F1 (0.94009)**: This represents a balanced metric that integrates both precision and recall. An F1 score near 1.0, as observed in our study, indicates that the model demonstrates a high level of accuracy and completeness in its responses, successfully managing the balance between precision and recall.

- **BERTScore Precision (0.91825)**: This score assesses the accuracy of the output generated by the model. A precision score of about 0.918 indicates that a high proportion of the responses from the model were relevant or correct in the context they were used.

- **BERTScore Recall (0.96299)**: This measures the model's ability to retrieve or gener-

ate all relevant instances or information. A recall score of about 0.963 suggests that the model successfully retrieved or generated almost all relevant information.

4.2.2 F1 Score and Exact Match

The F1 Score is a standard metric to evaluate the accuracy of a text generation model, especially when the model is used for classification tasks such as determining the relevancy or categorization of the generated text. F1 metric is useful when you need to balance the precision and recall of the model's predictions, which is often the case in natural language processing (NLP) applications. For our model, the average f1 score is 0.26125 and the Exact Match score is 0.00 for the exact match the tokens of the model's Generated text should exactly match the tokens of reference text then the Exact-Match score will be 100 otherwise it will be 0 as in our case.

Summary of Results:

The fine-tuned model demonstrated excellent performance, producing accurate and relevant responses to the musical instructions it received. It achieved impressive scores in various metrics, with a BERTScore F1 of 0.94009, which measures the overall quality of the generated text. The precision score was 0.91825, indicating the model's ability to produce relevant responses, and the recall score was 0.96299, reflecting the model's capability to retrieve all relevant content. These standard metrics measure the fine-tuned model's efficiency and accuracy in handling musical queries, making it a reliable and feasible tool in educational frameworks.

This success directly addresses our research question by describing how Large Language Models (LLMs) can be effectively used to understand and engage with diverse musical education contents, suggesting great opportunities for further exploration and development. This thesis approach provides a clear and sound path forward, showcasing the potential of integrating advanced fine-tuning techniques to create a more accurate, efficient, and context-aware model. This Fine-tuned model can be integrated with other LLMs for the music education framework.

/5 Discussion

The results from this study highlight the application of Large Language Models (LLMs) in musical education framework, specifically through their capacity to provide accurate responses to musical queries. This capability highlights the potential of Large Language Models to transform educational settings by offering personalized and precise assistance to learners. In the validation of the models' responses, it was observed that the model accurately generated responses for instructions that were already added within the instruction dataset. This accuracy shows the effectiveness of the fine-tuning process. However, introducing new instructions to the fine-tuned model and its outputs showed a decline in accuracy, marked by instances of hallucination. This study utilized specific datasets which may not represent all musical educational contexts. Additionally, the Large language Model's (LLM's) performance was only as good as the data it was trained on, which might limit its effectiveness in scenarios not covered during training. For example, the model response to the prompt "*Write a simple melody using C minor chord*" the model wrote a sequence of notes creating a melody while when we asked "*Write a simple melody using c minor in jazz style*" here the model was lacking its abilities.

Interpretation of Training and Evaluation and Analysis of Performance Metrics

The training and evaluation process shows a clear improvement in the loss over the training epochs, showing effective learning from our instruction dataset. At the beginning, the training loss was high, but it decreased by Epoch 1 as shown in the Figure 4.1, suggesting that the model was successfully adjusting its weights to minimize errors. This decrease in loss shows the model's ability to learn and adapt to the given musical instructions effectively. The evaluation of our fine-tuned model using BERTScore and F1 provides a comprehensive understanding of its performance. The BERTScore F1 of 0.94009 as shown in the Table 4.3 indicates a high level of accuracy and completeness in the model's responses. A precision

score of 0.91825 indicates that a significant majority of the model's outputs are relevant to the given prompts validated through the reference data. High precision means the model is effectively reducing false positives, ensuring that the responses generated are contextually appropriate and accurate. This is important in educational frameworks where precision is essential to avoid providing incorrect information to students. On the other hand, the BERTScore Recall of 0.96299 measures the proportion of relevant instances that have been retrieved over the total amount of relevant instances. A recall score of 0.96299 depicts that the model successfully generates almost all relevant information in its responses. High recall shows that the model is comprehensive in its output, capturing the full breadth of the relevant content needed to answer a query.

Addressing the Research Questions

In addressing RQ1, the study confirms that LLMs using the QloRA approach can be fine-tuned on specific musical data to increase their ability to understand and respond to a diverse range of musical queries, from basic theory questions to complex compositional feedback. This capability facilitates personalized learning by tailoring responses to fit the unique educational requirements and creative inputs of students in music education. This study [47] about Large Language Models (LLMs) like GPT and BERT has explained their ability to automate a variety of educational tasks across different categories such as labeling, assessment and grading, prediction, feedback, content generation, and recommendations and others.

In response to RQ2, one of the main challenges in designing a music education framework that provides personalized and adaptive learning experiences is the integration of Large Language Models (LLMs). These language models require continuous tuning with the updated and creative needs of music education. It is important to fine-tune LLMs with a wide range of music instruction datasets to ensure they can understand and respond to diverse musical queries and adapt to the changes within educational requirements over time.

Regarding RQ3, When developing an AI-based music education framework using LLMs, it's important to focus on several key design principles to ensure its effectiveness and user engagement. Firstly, interactivity is crucial, allowing students to receive immediate, personalized feedback. The system should adapt its content to individual learning styles and skill levels, promoting personalized learning. The framework should have the ability to allow students to experiment with creative music styles. As mentioned earlier integration is the main part of scalability also ethical considerations must address fairness in feedback across different musical genres and styles to prevent bias. The user interface should be intuitive, making the system accessible to all learners. Furthermore, the framework should include mechanisms for assessing and tracking student progress, also helping the student to provide feedback in their composition task or allow them to provide a MIDI file then the system will make suggestions which we will discuss in the future work section.

Regarding RQ4, user engagement and improvement in compositional skills can be measured using different metrics and techniques. One approach is to use pre and post-assessments to check students' compositional skills before and after using the music education framework.

The assessment will contain tasks such as making a piece of music, analyzing musical structures, or creating a melody using chords. Moreover, engagement metrics such as time spent on the educational platform, number of interactions with the framework, and completion rates of exercises can provide good statics on user engagement. Integrating a feature that allows users to review and reflect on their previous works and the feedback received can also help measure progress and engagement over time.

During the thesis project, we faced some technical challenges and resource limitations that affected the performance and completeness of the music education framework. The thesis topic was great for exploring the application of Large Language Models for music education while our unfamiliarity with music and music models was time-consuming as in limited time learning and implementing new models was challenging. However, we successfully laid down the process and implementation for fine-tuning the Large Language Model using PEFT QLoRA for downstream tasks. Moreover, we also face limited memory and GPU as pre-trained transformer-based Large Language Models are large and need high computation and large memory environments.

6

Conclusion & Future Work

In this study, we have explored the potential of Large Language Models (LLMs) within the area of music education, particularly highlighting the effectiveness of the Parameter Efficient Fine-Tuning QLoRA approach. This methodology has shown results in accurately interpreting and responding to musical queries, thereby enhancing educational practices. Despite these advancements, the study also identifies significant limitations, including the model's dependency on extensive and continuous training to manage diverse and evolving musical content. Additionally, potential biases in the training dataset could lead the model to favor certain music genres, styles, or cultures, which highlights the need for careful selection of training data.

6.1 Future Work

In future work, we suggest making an instruction dataset that includes a wide range of music theory, styles, and genres. A comprehensive dataset that covers all music theory will allow us to address the bias and hallucination issues of pre-trained models. Moreover, our fine-tuned model will be able to handle diverse musical queries and also able to generate more correct and reliable responses.

Our current fine-tuned model handles only text inputs and can generate text-based responses for music-related queries. To improve this music education framework we will use the MusicBERT model which can understand and process symbolic music data like MIDI formats. The MIDI format contains important information such as pitch, chords, notes, duration, and velocity of notes. Integrating MusicBERT will improve the adaptability and the performance of the music education framework allowing it to provide feedback and creative suggestion to the students' music compositions, hence improving their musical compositional skills.

Furthermore, LangChain, an open-source library, can be used to integrate our fine-tuned model with MusicBERT. LangChain will enable a more efficient and effective educational tool that combines the strengths of both fine-tuned models and MusicBERT. It will create a pipeline between these two models where MusicBERT will process the symbolic music data like MIDI and the output will be used by a fine-tuned model to generate educational content and provide feedback and creative suggestions on compositions. Integrated systems can be used to provide real time feedback and suggestions for students' MIDI compositions based on technical accuracy, styles, notes duration, velocity, and other features processed by MusicBERT.

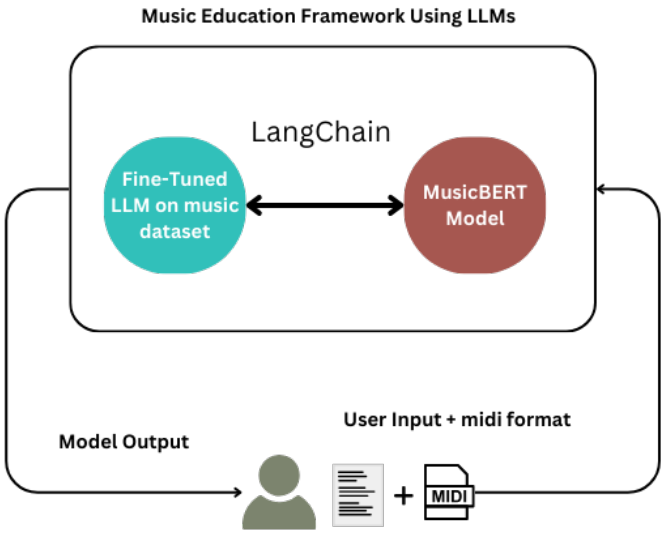


Figure 6.1: Music education framework using LLMs

Bibliography

[1] Packt Publishing. *Large Language Models (LLMs) in Education*. 2023. URL: https://www.packtpub.com/article-hub/large-language-models-llms-in-education (visited on 03/31/2024).

[2] Wensheng Gan et al. "Large Language Models in Education: Vision and Opportunities." In: *Not Specified* (2023).

[3] Edward J Hu et al. "Lora: Low-rank adaptation of large language models." In: *arXiv preprint arXiv:2106.09685* (2021).

[4] Tim Dettmers et al. "Qlora: Efficient finetuning of quantized llms." In: *Advances in Neural Information Processing Systems* 36 (2024).

[5] Zihao Deng et al. *MusiLingo: Bridging Music and Text with Pre-trained Language Models for Music Captioning and Query Response*. 2023. arXiv: 2309.08730 [eess.AS].

[6] Ping-ping Li and Bin Wang. "Artificial intelligence in music education." In: *International Journal of Human–Computer Interaction* (2023), pp. 1–10.

[7] Xiaofei Yu et al. "Developments and applications of artificial intelligence in music education." In: *Technologies* 11.2 (2023), p. 42.

[8] Elastic. *What Are Large Language Models?* Accessed: 2024-03-31. 2023. URL: https://www.elastic.co/what-is/large-language-models (visited on 03/31/2024).

[9] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[10] Luciano Floridi and Massimo Chiriatti. "GPT-3: Its Nature, Scope, Limits, and Consequences." In: *Minds & Machines* 30 (2020), pp. 681–694. DOI: 10.1007/s11023-020-09548-1. URL: https://doi.org/10.1007/s11023-020-09548-1.

[11] Jean-Pierre Briot. "From artificial neural networks to deep learning for music generation: history, concepts and trends." In: *Neural Computing and Applications* 33 (2021), pp. 39–65. DOI: 10.1007/s00521-020-05399-0. URL: https://doi.org/10.1007/s00521-020-05399-0.

[12] Shulei Ji, Xinyu Yang, and Jing Luo. "A survey on deep learning for symbolic music generation: Representations, algorithms, evaluations, and challenges." In: *ACM Computing Surveys* 56.1 (2023), pp. 1–39.

[13] Davide. *What is ABC?* [accessed 25-April-2024]. 2023. URL: https://notabc.app/abc/what/.

[14] Ruibin Yuan et al. *ChatMusician: Understanding and Generating Music Intrinsically with LLM*. 2024. arXiv: 2402.16153 [cs.SD].

[15] Ilaria Manco et al. "Muscaps: Generating captions for music audio." In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–8.

[16] Qingqing Huang et al. *MuLan: A Joint Embedding of Music Audio and Natural Language*. 2022. arXiv: 2208.12415 [eess.AS].

[17] Yizhi Li et al. *MERT: Acoustic Music Understanding Model with Large-Scale Self-supervised Training*. 2024. arXiv: 2306.00107 [cs.SD].

[18] Wei-Lin Chiang et al. *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality*. Mar. 2023. URL: https://lmsys.org/blog/2023-03-30-vicuna/.

[19] Shulei Ji, Xinyu Yang, and Jing Luo. "A Survey on Deep Learning for Symbolic Music Generation: Representations, Algorithms, Evaluations, and Challenges." In: *ACM Comput. Surv.* 56.1 (Aug. 2023). ISSN: 0360-0300. DOI: 10.1145/3597493. URL: https://doi.org/10.1145/3597493.

[20] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. *Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription*. 2012. arXiv: 1206.6392 [cs.LG].

[21] Cheng-Zhi Anna Huang et al. "Music transformer: Generating music with long-term structure (2018)." In: *arXiv preprint arXiv:1809.04281* (2018).

[22] Yu-Siang Huang and Yi-Hsuan Yang. *Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions*. 2020. arXiv: 2002.00212 [cs.SD].

[23] Shayan Dadman et al. "Toward Interactive Music Generation: A Position Paper." In: *IEEE Access* 10 (2022), pp. 125679–125695. DOI: 10.1109/ACCESS.2022.3225689.

[24] Jean-Pierre Briot and François Pachet. "Deep learning for music generation: challenges and directions." In: *Neural Computing and Applications* 32.4 (Oct. 2018), pp. 981–993. ISSN: 1433-3058. DOI: 10.1007/s00521-018-3813-6. URL: http://dx.doi.org/10.1007/s00521-018-3813-6.

[25] Mingliang Zeng et al. *MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training*. 2021. arXiv: 2106.05630 [cs.SD].

[26] Mathav Raj J et al. *Fine Tuning LLM for Enterprise: Practical Guidelines and Recommendations*. 2024. arXiv: 2404.10779 [cs.SE].

[27] Najeeb Nabwani NLP Research Engineer. *Fine-Tuning, PEFT, Prompt Engineering, and RAG: Which One Is Right for You?* Accessed: 2024-05-09. 2023. URL: https://deci.ai/blog/fine-tuning-peft-prompt-engineering-and-rag-which-one-is-right-for-you/.

[28] Lingling Xu et al. *Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment*. 2023. arXiv: 2312.12148 [cs.CL].

[29] Sourab Mangrulkar and Sayak Paul. *PEFT: Parameter-Efficient Fine-Tuning of Billion-Scale Models on Low-Resource Hardware*. https://huggingface.co/blog/peft. Accessed: 2024-05-09. Feb. 2023.

[30] Benoit Jacob et al. *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference*. 2017. arXiv: 1712.05877 [cs.LG].

[31] Meta AI. *Retrieval Augmented Generation: Streamlining the creation of intelligent natural language processing models*. https://ai.meta.com/blog/retrieval-augmented-generation-streamlining-the-creation-of-intelligent-natural-language-processing-models/. Accessed: date-of-access. 2020.

[32] Yunfan Gao et al. *Retrieval-Augmented Generation for Large Language Models: A Survey*. 2024. arXiv: 2312.10997 [cs.CL].

[33] Mudassar Amin. *Thesis Project*. Accessed: 2024-05-14. 2024. URL: https://github.com/mudassar-amin/thesis_project.

[34] Jason Wei et al. "Finetuned Language Models are Zero-Shot Learners." In: *International Conference on Learning Representations*.

[35] Mike Conover et al. "Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM." In: (Apr. 2023).

[36] Rohan Taori et al. "Alpaca: A Strong, Replicable Instruction-Following Model." In: (2023).

[37] Vipula Rawte, Amit Sheth, and Amitava Das. "A survey of hallucination in large foundation models." In: *arXiv preprint arXiv:2309.05922* (2023).

[38] *OpenAI Pricing*. https://openai.com/pricing. Accessed: April 10, 2024.

[39] Authors. "Llama 2: Open Foundation and Fine-Tuned Chat Models." In: *AI at Meta* (2023). Accessed: date. URL: https://ai.meta.com/research/publications/llama-2-open-foundation-and-fine-tuned-chat-models/.

[40] Bharat Runwal, Tejaswini Pedapati, and Pin-Yu Chen. "From PEFT to DEFT: Parameter Efficient Finetuning for Reducing Activation Density in Transformers." In: *arXiv preprint arXiv:2402.01911* (2024).

[41] Hugging Face. *Quantization - PEFT*. https://huggingface.co/docs/peft/main/en/developer_guides/quantization. Accessed: 2024-04-10.

[42] Hugging Face. *Adapters in PEFT*. https://huggingface.co/docs/peft/main/en/conceptual_guides/adapter. Accessed: 2024-04-11. 2023.

[43] Sujatha Mududla. *Difference Between Trainer Class and SFTTrainer (Supervised Fine-Tuning Trainer) in Hugging Face*. 2023.

[44] First name Author's Last name. *Title of the Article*. https://data-amplifier.beehiiv.com/p/llms-fine-tuning-and-model-evaluation. Accessed: date-of-access. 2023.

[45] Julian Risch et al. *Semantic Answer Similarity for Evaluating Question Answering Models*. Available at mailto:julian.risch@deepset.ai, timo.moeller@deepset.ai, julian.gutsch@deepset.ai, malte.pietsch@deepset.ai. deepset, 2023.

[46] Tianyi Zhang* et al. "BERTScore: Evaluating Text Generation with BERT." In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=SkeHuCVFDr.

[47] Lixiang Yan et al. "Practical and ethical challenges of large language models in education: A systematic scoping review." In: *British Journal of Educational Technology* 55.1 (Aug. 2023), pp. 90–112. ISSN: 1467-8535. DOI: 10.1111/bjet.13370. URL: http://dx.doi.org/10.1111/bjet.13370.

[48] Confident AI. *The Ultimate Guide to Fine-Tune LLaMA 2 with LLM Evaluations*. https://www.confident-ai.com/blog/the-ultimate-guide-to-fine-tune-llama-2-with-llm-evaluations. Accessed: 2024-04-10. 2023.

[49] Jacob Holster. *Augmenting Music Education Through AI: Practical Applications of ChatGPT*. Oct. 2023. DOI: 10.13140/RG.2.2.35958.57923/1.

A

Appendix

Task Description

This appendix includes the original task description for this thesis. The following pages present the full document as it was received at the start of the project period containing problem description, research objectives, and questions.

UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of Engineering Science and Technology
Department of Computer Science and Computational Engineering
UiT - The Arctic University of Norway

Development of a Music Education Framework Using Large Language Models (LLMs)

Mudassar Amin

Thesis for Master of Science in Technology / Sivilingeniør

Background

Traditional music composition teaching methods are often time-consuming and require a considerable investment of time and resources. Many aspiring composers encounter difficulties such as insufficient access to expert guidance, expensive private tuition, and a lack of immediate feedback on their work. Conventional classroom settings and private lessons may not always accommodate individual learning styles or offer the level of interaction needed to inspire creativity and maintain interest.

Furthermore, learning music can be challenging for beginners due to the complexity of music theory and compositional techniques. Consequently, beginners may get discouraged from pursuing their interest in music composition. A learning platform that can adapt to the learner's knowledge and skills is required to tackle this issue. Such a platform should provide a personalized learning path that aligns with the learner's objectives and interests.

Problem description

The current advancement of Large Language Models (LLMs) has made it possible to create text resembling human language. This development brings great potential for personalized learning experiences and improved music education. Indeed, it can provide more accessible and efficient ways to learn music.

The challenge is to develop an AI-powered tool that can effectively address the difficulties faced in music education. The tool should be designed to provide an inclusive, engaging, and customized learning experience for music composition. It must be able to comprehend and respond to various musical queries and inputs, offer helpful feedback, and guide users through the learning process. The tool should act as a virtual mentor, capable of honing the user's compositional skills.

Therefore, this thesis aims to develop an interactive framework that uses LLMs to make music education more accessible to everyone. This framework is meant to meet the needs of both music students and educators. It attempts to alleviate the limitations of traditional music education. Indeed, it seeks to introduce a teaching method that incorporates effectiveness, accessibility, and engagement.

Scope and limitations

- Focus on the academic part and research problems.
- Implementing a commercial product is out of scope.
- Proof-of-concept implementation of algorithms are expected to emerge.

Research questions

- How can Large Language Models (LLMs) be effectively utilized to understand and respond to diverse musical queries and inputs in music education?
- What are the main challenges in designing a framework for music education that provides personalized and adaptive learning experiences?
- What are the key design principles when developing an AI-based music education framework?
- How can user engagement and improvement in compositional skills be measured for such a framework?

Research Objectives

- Review the current state of research and identify potential gaps in the application of Large Language Models (LLMs) in music education.
- Conduct a feasibility study on using LLMs for interpreting and responding to musical inquiries, specifically focusing on technical limitations and opportunities.
- Develop a music education framework that utilizes LLMs to provide feedback and ideas on simple music composition tasks, demonstrating LLMs' potential in targeted educational contexts.
- Evaluate the effectiveness of the framework by analyzing the quality of LLM-generated feedback on relevance and educational value.

Dates

Date of distributing the task: <08.01.2024>

Date for submission (deadline): <21.05.2024>

Contact information

Candidate

Mudassar Amin
mam065@uit.no

Supervisors at UiT-IDBI

Bernt Arild Bremdal
bernt.a.bremdal@uit.no

Shayan Dadman
shayan.dadman@uit.no

Kalyan R Ayyalasomayajula
kalyan.r.ayyalasomayajula@uit.no

General information

This master thesis should include:

- ❄ Preliminary work/literature study related to actual topic
  - A state-of-the-art investigation
  - An analysis of requirement specifications, definitions, design requirements, datasets, given standards or norms, guidelines and practical experience etc.
  - Description concerning limitations and size of the task/project
  - Estimated time schedule for the project/ thesis
- ❄ Selection & investigation of actual materials
- ❄ Development (creating a model or model concept)
- ❄ Experimental work (planned in the preliminary work/literature study part)
- ❄ Suggestion for future work/development

Preliminary work/literature study

After the task description has been distributed to the candidate a preliminary study should be completed within 3 weeks. It should include bullet points 1 and 2 in "The work shall include", and a plan of the progress. The preliminary study may be submitted as a separate report or "natural" incorporated in the main thesis report. A plan of progress and a deviation report (gap report) can be added as an appendix to the thesis.

In any case the preliminary study report/part must be accepted by the supervisor before the student can continue with the rest of the master thesis. In the evaluation of this thesis, emphasis will be placed on the thorough documentation of the work performed.

Reporting requirements

The thesis should be submitted as a research report and could include the following parts; Abstract, Introduction, Material & Methods, Results & Discussion, Conclusions, Acknowledgments, Bibliography, References and Appendices. Choices should be well documented with evidence, references, or logical arguments.

The candidate should in this thesis strive to make the report survey-able, testable, accessible, well written, and documented.

Materials which are developed during the project (thesis) such as software / source code or physical equipment are considered to be a part of this paper (thesis). Documentation for correct use of such information should be added, as far as possible, to this paper (thesis).

The text for this task should be added as an appendix to the report (thesis).

General project requirements

If the tasks or the problems are performed in close cooperation with an external company, the candidate should follow the guidelines or other directives given by the management of the company.

The candidate does not have the authority to enter or access external companies' information system, production equipment or likewise. If such should be necessary for solving the task in a satisfactory way a detailed permission should be given by the management in the company before any action are made.

Any travel cost, printing and phone cost must be covered by the candidate themselves, if and only if, this is not covered by an agreement between the candidate and the management in the enterprises.

If the candidate enters some unexpected problems or challenges during the work with the tasks and these will cause changes to the work plan, it should be addressed to the supervisor at the UiT or the person which is responsible, without any delay in time.

The candidate is required to demonstrate continuous progress in developing and completing the work. Continuous progress is defined as consistent, substantive advancement in research, analysis, writing, and other activities directly related to fulfilling the thesis requirements determined by the supervisory committee.

If the candidate fails to demonstrate continuous progress, as evidenced by regularly requesting help without showing substantive independent effort or failing to engage with the supervisors or the thesis work, the supervisors may limit or withdraw support and assistance.

Submission requirements

This thesis should result in a final report with an electronic copy of the report including appendices and necessary software, source code, simulations and calculations. The final report with its appendices will be the basis for the evaluation and grading of the thesis. The report with all materials should be delivered according to the current faculty regulation. If there is an external company that needs a copy of the thesis, the candidate must arrange for it. A standard front page, which can be found on the UiT internet site, should be used. Otherwise, refer to the "General guidelines for thesis" and the subject description for master thesis.

The supervisor(s) should receive a copy of the the thesis prior to submission of the final report. The final report with its appendices should be submitted no later than the decided final date.