Enhancing Text-Audio Generation By Genre Classification And RAG

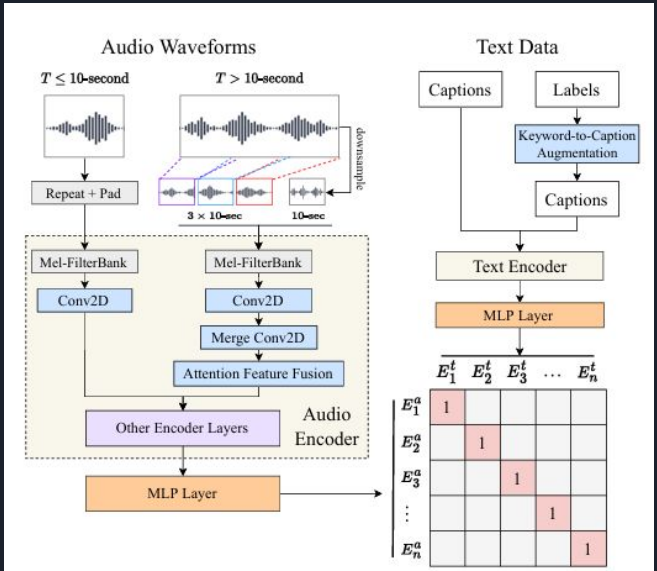
Chris Wang, David He, Evelyn Zhu, Yanice Yin

Contents:

Intro & Problem Statement

Background: Music Gen

MusicGen is Language Model(LM) that operates over music representation while being conditioned on textual description



Problems:

- Music data are highly unstructured
- Text input length affects performance
- Variety of text affects performance

Our Solution:

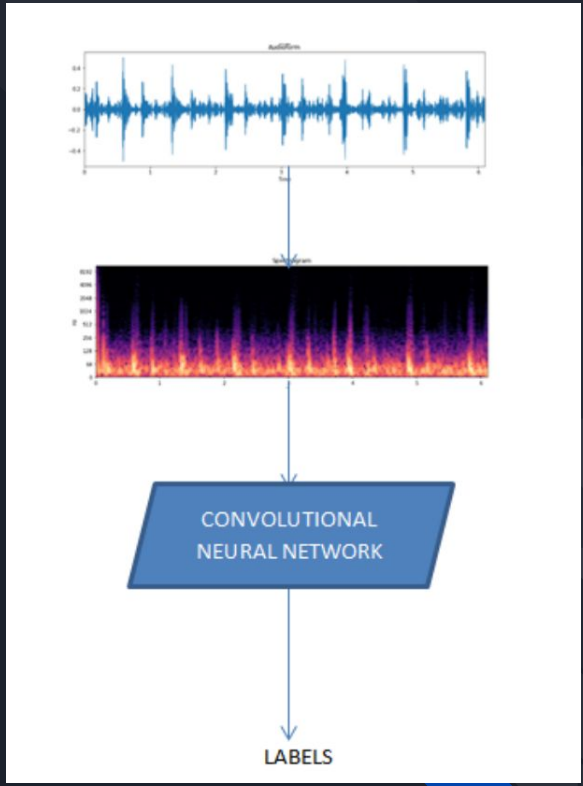
1. Train Classifier for Different Music Characteristics
2. Build Characteristics-Specific Database for Generation
3. Build RAG Pipeline to map text input to music reference during generation

Phase 1: Music Classification

Understanding Classifier:

A spectrogram is the Short Time Fourier Transform (STFT) of an audio signal. They are images showing time and frequency components of an audio signal.

Data of music waves are first converted into spectrogram. Then CNN is applied to spectrogram in order to predict the label.

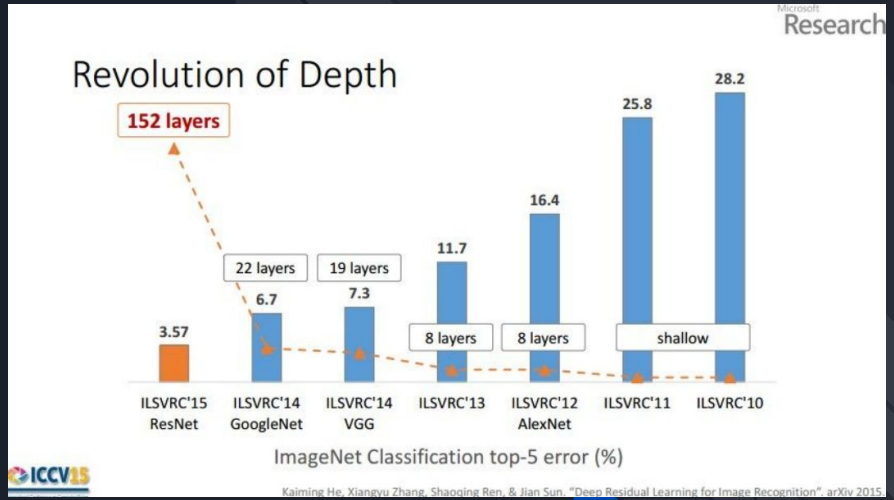


Define Testing Dataset:

We primarily used GTZAN for classification testing. GTZAN is a data set composed of 1000 audios with 30 second in length, labeled by one of ten genres .

Comparing Classifier:

We compared different CNN architectures and selected ResNet50, which achieved the highest genre classification accuracy of 70%.



Phase 2: Data Collection and Classification

Music Pre-Processing

We created pipeline to convert vastly available .mp3 files into .wav files, in order to build our own database.

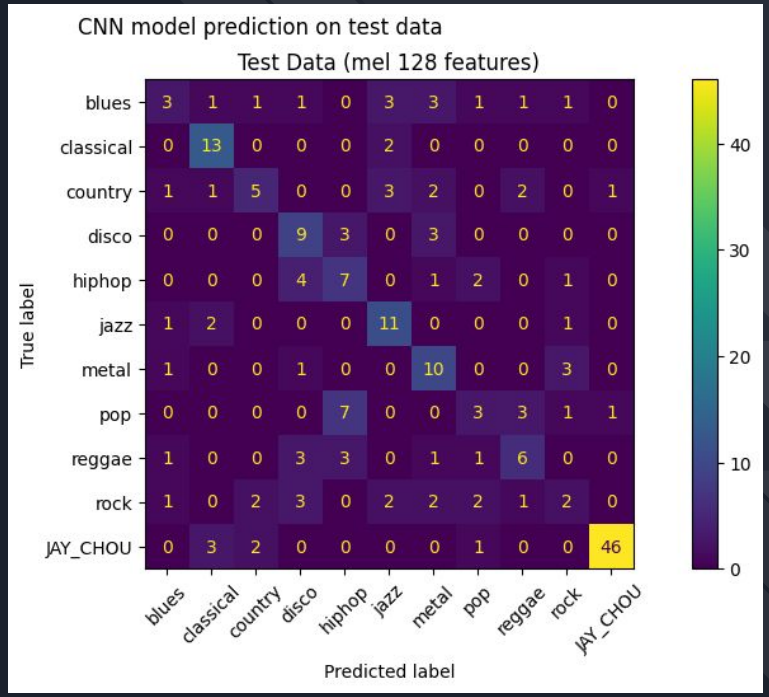
```python
def convert_mp3_to_wav(input_path, output_path):
    try:
        (
            ffmpeg
            .input(input_path)
            .output(output_path, format='wav', acodec='pcm_s16le', ac=1, ar='22050')
            .run(overwrite_output=True)
        )
        print(f"Successfully converted {input_path} to {output_path}")
    except ffmpeg.Error as e:
        print(f"Failed to convert {input_path}. Error: {str(e)}")

def batch_convert_mp3_to_wav(source_folder, output_folder):
    # Create the output folder if it doesn't exist
    os.makedirs(output_folder, exist_ok=True)

    # Loop over all files in the source folder
    for file_name in os.listdir(source_folder):
        if file_name.lower().endswith('.mp3'):
            input_file = os.path.join(source_folder, file_name)
            output_file = os.path.join(output_folder, file_name[:-4] + '.wav')
            convert_mp3_to_wav(input_file, output_file)
```

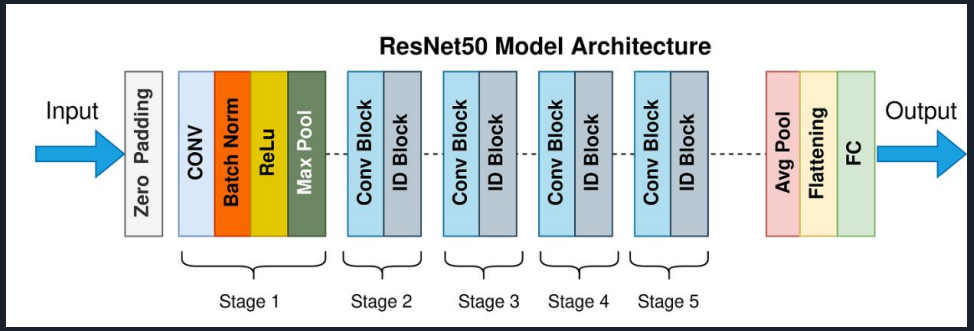
Classification Testing

The Classifier is capable of identifying artist's personal style.



Classifier Finalization

We further advanced the accuracy of classifier by fine tuning ResNet50 and incorporating GoogleNet.



Classification Testing

Successfully classified JAY CHOU pure music with other types of music, overall 0.78 accuracy, 0.96 accuracy achieve for JAY CHOU specific music type

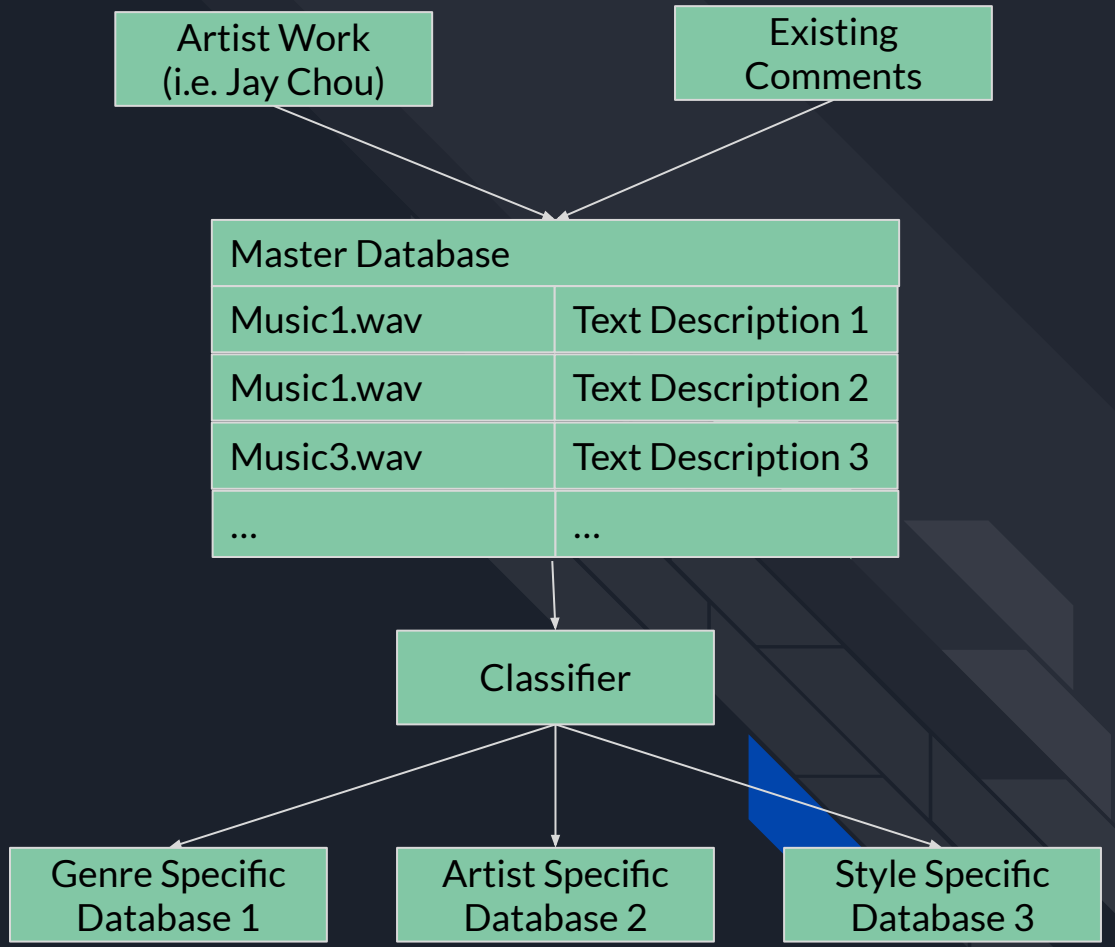
	precision	recall	f1-score
0	0.56	0.60	0.58
1	0.82	0.93	0.87
2	0.71	0.67	0.69
3	0.87	0.87	0.87
4	0.93	0.93	0.93
5	0.80	0.80	0.80
6	0.79	0.73	0.76
7	0.90	0.60	0.72
8	0.73	0.73	0.73
9	0.53	0.60	0.56
10	0.96	1.00	0.98
accuracy			0.81
macro avg	0.78	0.77	0.77
weighted avg	0.82	0.81	0.81

Phase 3: Retrieval Augmented Generation

Data Annotation & Organization

We leveraged the existing comments on musics to create music - description pairs and store them into a master database.

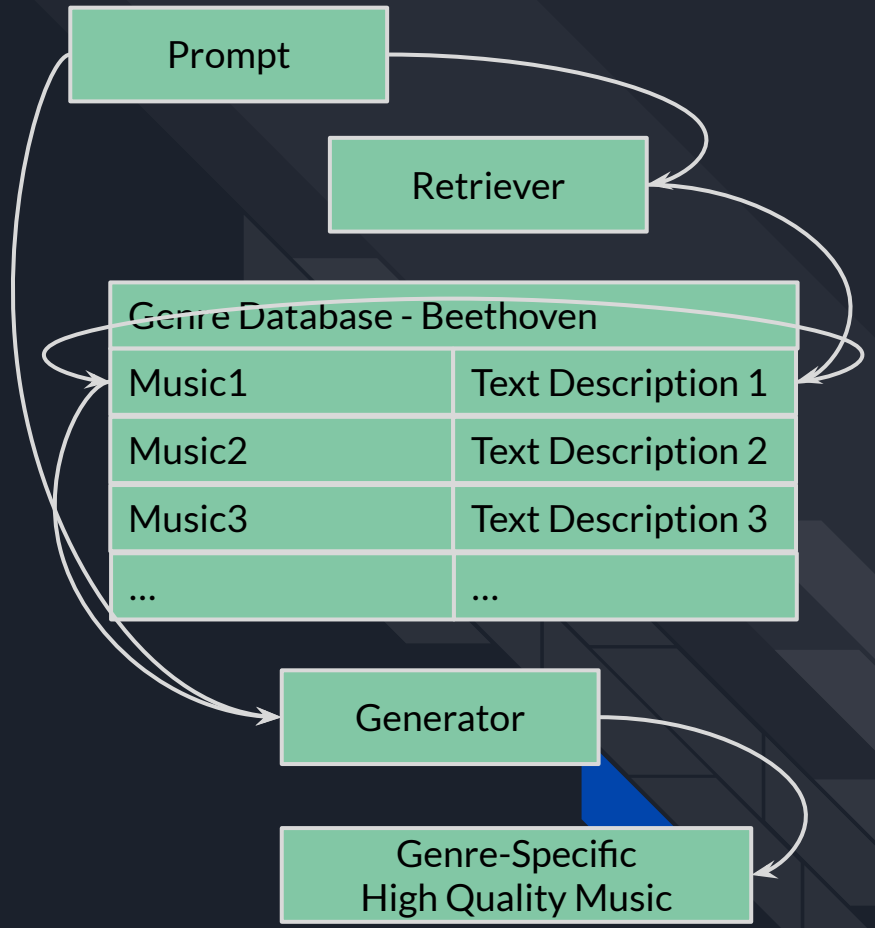
We applied classifier to the master database, in order to get databases with specific genre, artist, or style.



Retrieval Augmented Generation

We employed retriever to obtain the most relevant music annotation from database.

We map the annotation to the existing well-formulated music, as an input to our generator.



Thank You