Journey Through Music and AI

Introduction

I began this project with a vision to revolutionize the music industry using AI. Although I couldn't achieve everything I set out for, the journey taught me valuable lessons. From creating vocals and lyrics to generating soundtracks, each step brought its challenges, laughter, and learning.

Initial Goals

- Develop a model for human-like vocal synthesis.
- Implement an advanced Hindi and English lyrics completion system.
- Explore soundtrack creation for varying genres.
- Understand preprocessing workflows from Jukebox and integrate VQVAE techniques.

Notebook 1: Umeed.ipynb

Objective: Create vocals using mBART for singing synthesis.

This notebook uses the mBART architecture for generating vocals. Here's the workflow in detail:
1. **Input Processing**: Lyrics are tokenized using the SentencePiece tokenizer. Tokens are padded to handle varying lengths.
2. **Model Architecture**: The mBART encoder-decoder structure processes input embeddings and generates spectrogram-like outputs.
3. **Training Details**: Limited to the NUS-48E dataset, results were constrained by small data and high resource needs.

Problems:-

The dataset used is NSU48E which contains phonemes which is used for training on music data as its better to work on phonemes than lyrics for generating vocals but some phonemes are going out of vocabulary of mbart tokenizer so its creating a problem

Now I am trying to incorporate lyrics instead of phonemes because phonemes are creating a problem in training but its producing audio but only disturbances are produced

NOTE: For both the model architecture Umeed and Ajeeb_Umeed ipynb(mentioned below) they are producing disturbances but they are learning as in case of prompt like rock etc the sound produced is quite loud while if soft is mentioned or soothing is mentioned sound is quite low Also since both of them are trained on NSU48E dataset which contain just 48 files the models are not able to train themselves properly and for it we need a large dataset like FSD50k but due to the large amount of space occupied by it and large amount of training time presently it's not possible to use it.

Notebook 2: Ajeeb_Umeed.ipynb

Objective: Build an advanced vocal synthesis model.

It uses Neural Source Filter model which are designed specifically for singing synthesis

- **Embedding Layer**:
  - Converts each character in the input lyrics (mapped using `vocab`) into a fixed-length vector.
  - This gives the network a way to represent text as numbers that it can work with.

- **LSTM (Long Short-Term Memory)**:
  - A bidirectional LSTM processes the embedded text.
  - Bidirectional means the LSTM considers both past and future context in the sequence.
  - Outputs a hidden state for each character that incorporates contextual information from the entire sequence.

Why this approach?

- **Capturing Temporal Information**: Using LSTM in the text encoder helps capture the sequential nature of lyrics. It can handle dependencies in the lyrics that are important for generating coherent and meaningful singing.
- **Pitch Generation**: Separately modeling pitch allows the model to control the melodic contour, which is a crucial aspect of singing synthesis. The pitch values can be interpolated with lyrics to create a natural singing voice.

- **Output Alignment**: By projecting the encoded features to match the mel spectrogram length and using interpolation, the model can produce output sequences that align well with the desired audio length. This approach is more flexible compared to fixed-length output prediction models.

The model used here is relatively simply and I wanted to implement a more advance model which I have written in the ipynb file in comments which uses transformer encoders etc to produce good music but error were coming due to dynamic output

Limitations and Future Work

- Resource limitations prevented large-scale training.
- Padding issues caused instability during vocal synthesis.
- Future improvements could include larger datasets and alternative architectures like Tacotron.

Notebook 3: English Lyrics Completion

Objective: Implement an English lyrics completion system that utilizes mBart.
This notebook explores natural language processing (NLP) methods to predict and complete lyrics. The model created works considerably fine and is producing good lyrics completion like

```
Prompt: I remember when we broke up the first time
Completion:  In the darkest little paradise
```

```
Prompt: Saying goodbye is death by a thousand cuts
Completion:  All I know is that you held the door
```

```
Prompt: And I can go anywhere I want
Completion:  Cause I'm not the kind of girl
```

Key Steps:
1. **Preprocessing:** The text corpus was tokenized and formatted for mBart fine-tuning.
2. **Model Architecture**: mBart's transformer structure was utilized for its ability to capture contextual relationships.
3. **Training**: The model was fine-tuned on a dataset of English lyrics (TaylorSwift Dataset), optimizing for the next-word prediction.

Challenges:
- Fine-tuning mBart required significant computational resources.
- Dataset limitations affected the model's ability to generalize.

Notebook 4: Jukebox Preprocessing (Learn.ipynb)

Objective: Preprocess audio data for integration with Jukebox, focusing on mel-spectrogram generation.

Key Steps:
1. **Data Preparation**: Audio files were collected and segmented.
2. **Feature Extraction**: Mel-spectrograms were generated for each audio file.
3. **Integration**: Preprocessed data was structured to align with Jukebox's VQVAE model requirements. Tried to mimic juke box.

Challenges:
- Processing large audio datasets required high storage and computation.
- Mel-spectrogram extraction was time-intensive for large audio files.

Results and Metrics:

Audio Generation (Umeed and Ajeeb_Umeed.ipynb)

Loss Functions Used:

1. **STFT Loss:**
   - **Formula:**

$$X(\omega, \tau) = \int_{-\infty}^{\infty} x(t) w(t - \tau) e^{-j\omega t} dt$$

- w(t): Window function (e.g., Hamming, Hann).
- x(t): Input signal.
- τ: Time shift parameter.
- ω: Frequency.

$$\text{STFT Loss} = \|\text{STFT}(\text{pred}) - \text{STFT}(\text{target})\|_1$$

   - **Reason for Use:** Unlike simple waveform-based losses like Mean Squared Error (MSE), STFT Loss analyzes audio in the frequency domain. This allows it to evaluate differences in tonal quality and spectral characteristics, which are crucial for generating realistic audio.

Training Observations:

- Despite training, both models encountered a plateau in performance after 12 epochs, with minimal improvements in loss values.

Epoch-wise Loss Reduction:

Epoch	Loss
1	3750.6873
5	421.2151
10	235.5260
15	235.4996
18	235.3468

Challenges:

The plateau in loss values can be attributed to:

1. **Dataset Size:** The NSU48E dataset was limited to 48 files, restricting the model's learning capabilities.

Lyrics Completion (English Lyrics Completion.ipynb):

- **Model Architecture**:
  - The **mBART transformer** was fine-tuned for this task.
  - The model used a sequence-to-sequence approach where input lyrics were mapped to their next lines.
- **Training Setup**:
  - **Optimizer**: AdamW with a learning rate of 2×10−42 \times 10^{-4}2×10−4.
  - **Batch Size**: 4.
  - **Epochs**: 5.
  - **Validation Check**: At 25% intervals during training.
- **Output Examples**:
  - The model was tested with various prompts, and the completions showed contextual relevance:
    - **Prompt**: *I remember when we broke up the first time*
      Completion: *In the darkest little paradise*

Results:

The generated completions demonstrate that the model captures the lyrical style and context well. However, quantitative metrics like BLEU or Perplexity were not computed in this notebook, and evaluation focused primarily on qualitative assessments.

Other Works and Learnings:

I worked on a model for creating lyrics in devnagri language and also lyrics completion in hindi in English font by using gpt2 architecture only but it failed and at that moment I understood making AI understand hindi is really difficult.

Problems Faced

1. **Resource Requirements**: Large-scale training necessitates extensive computational resources.
2. **Dataset Size**: Audio datasets occupy significant storage space and require preprocessing for effective model training.
3. **Time Constraints**: Both training and preprocessing are time-intensive, particularly for audio data.
4. **Failed Attempt**: A model for Hindi lyrics completion using IndicBERT failed due to lack of dataset diversity and resource constraints. (Indic-bert)

Note: Due to my love for hindi songs I thought of doing hindi lyrics completion but failed in it but I tried my best creating models for it and they are producing lyrics but absolute rubbish.

Future Vision

1. Enhanced Models: Explore advanced architectures like Tacotron and WaveNet.
2. Expanded Datasets: Leverage larger, diverse datasets for improved performance.
3. Optimization: Develop efficient preprocessing pipelines to reduce time and resource consumption.
4. Cross-Lingual Models: Build robust systems for multilingual lyrics generation and completion. Although it's a difficult task as transfer learning can't be implemented due to unavailability of the models and making the things from complete basic.

Note: The code may look like ChatGPT but it's not due to my laziness to write code but the code I wrote is uploaded a no. of times on ChatGPT and Claude to optimize the code, solve errors and making improvements etc.

Conclusion

This project has been a journey of learning and discovery. While challenges like resource limitations and dataset constraints persist, the insights gained pave the way for future innovations in AI-driven music generation.

All dataset are available on kaggle except NSU48E whose link is there in G-Drive

https://drive.google.com/drive/folders/12pP9uUl0HTVANU3IPLnumTJiRjPtVUMx?usp=sharing