Proceedings of 15th IOE Graduate Conference
Peer Reviewed
Year: 2024 Month: May Volume: 15
ISSN: 2350-8914 (Online), 2350-8906 (Print)

Mode-Based Real Time Music Generation Using LSTM and GAN

Prabin Bohara [a], Prabin Sharma Poudel [b], Raj Kumar Dhakal [c], Sajjan Acharya [d], Shikhar Bhattarai [e]

[a, b, c, d, e] *Thapathali Campus, IOE, Tribhuvan University, Nepal*
[a] prabinbohara10@gmail.com, [b] prabinsharmapoudel@gmail.com, [c] rajdhakal.404@gmail.com, [d] sajjanacharya11@gmail.com, [e] shikhar28@tcioe.edu.np

Abstract
Composing music in a specific mode presents a daunting challenge due to the vast number of potential pitch and chord permutations, making it a tedious and time-consuming process. In this paper, we propose an advanced music generation system that addresses this issue by leveraging mode classification and separate training for accurate sequence prediction, ensuring compliance with music theory principles. Mode classification is achieved using the Hamming distance, a robust dissimilarity metric that enables precise identification of the underlying mode within a given dataset. By employing the concept of generative network whose core has been powered by Long Short-Term Memory (LSTM) models, we generate sequences of new pitch combinations and chord progressions. Incorporating a discriminative network in the system enhances the authenticity and quality of the generated compositions. The successful development of this system represents a significant contribution to the field of music generation, with the potential for further enhancements through the incorporation of advanced music theory concepts and rigorous validation techniques.

Keywords
Discriminator, Generator, Hamming distance, LSTM, MIDI, Modes, Pitch

1. Introduction

Understanding the properties of music is essential for a comprehensive comprehension of this intricate art form. This paper provides a concise overview of key music properties, covering notes, octaves, scales, chords, tempo, time signatures, intervals, tonic, dominant, sub-dominant, and shifts in notes.

Notes are the fundamental building blocks of music, representing specific pitches and durations. Octaves divide the musical range into distinct sections, with each octave comprising a set of twelve notes. Scales, on the other hand, establish tonal frameworks and create unique musical atmospheres by organizing notes in specific patterns. An interval is the pitch difference between two notes; Tonic, Dominant, and Sub-dominant specify the context of the interval. Meanwhile, Tempo and time signatures capture the rhythmic aspects of music. By delving into these musical properties, one can unravel the intricacies of composition and performance.

In the realm of music, modes encompass the ordering and arrangement of musical notes, shaping the tonality and creating distinct musical atmospheres. While the terms “mode" and “scale" are sometimes used interchangeably, they carry different meanings. A scale represents a sequence of notes organized in ascending or descending order, whereas a mode signifies a specific way of employing a scale within a musical context. Modes go beyond mere scales in terms of application, embodying a unique approach to utilizing notes to evoke a particular musical mood.

Modes, often referred to as “tenors," serve as tonal centers around which melodies revolve. They have been utilized for centuries in music composition, with the Major and Minor modes being the most prevalent. Modes are powerful tools for conveying diverse emotional landscapes. For example, the Major mode imparts a sense of brightness and cheerfulness, while the Minor mode evokes a mood of melancholy and introspection. Descriptions of musical tonality frequently involve referencing the key and its corresponding mode. For instance, if a piece is described as “E Aeolian," it signifies that the composition is in the key of ‘E’ as its root note or home, and the tonality of the piece is characterized as sombre and brooding.

There exist seven fundamental modes in music, each of which relates to the Major scale. These modes exhibit specific alterations to notes, introducing flattened or sharpened tones in comparison to the Major scale. The following modes exemplify this:

1. Ionian mode: 1, 2, 3, 4, 5, 6, 7
2. Lydian mode: 1, 2, 3, 4#, 5, 6, 7
3. Mixolydian mode: 1, 2, 3, 4, 5, 6, b7
4. Dorian mode: 1, 2, b3, 4, 5, 6, b7
5. Aeolian mode: 1, 2, b3, 4, 5, b6, b7
6. Phrygian mode: 1, b2, b3, 4, 5, b6, b7
7. Locrian mode: 1, b2, b3, 4, b5, b6, b7

Various other modes can exist, and they can differ from one part of the world to another. For instance, Eastern music can have different modes such as Bhairav, Marva, and Todi. Each mode incorporates distinctive alterations to certain notes, leading to the creation of unique tonal qualities and emotional characteristics. By understanding and harnessing the power of modes, composers can craft music that resonates with specific moods and atmospheres, allowing for rich and evocative musical expressions.

Mode-based music composition requires one to figure out a

way to classify the collected datasets into established modes, either by an algorithm of some kind or through an extensive training. The former method, with the aid of music theory, turns out to be more effective than the latter. Thus, the authors have employed an effective algorithm for mode classification, whose correctness is verified by the authors. In the subsequent phase of music generation a Deep Learning pipeline is employed.

2. Related Works

The authors of the paper by Google Research titled “Multi-instrument Music Synthesis with Spectrogram Diffusion" [1] employs a two-stage process, converting MIDI to spectrograms using an encoder-decoder transformer and then generating audio from the spectrograms using a convolution spectrogram inversion network in real-time.

An application is introduced in “An interactive music infilling interface for pop music composition," [2] to facilitate pop music composition with user control utilizing melody, bass, and harmony tracks.

The use of genetic algorithms for jazz melody generation is explored in “GenJam: A Genetic Algorithm for Generating Jazz Solos." [3] that employs genetic operators like crossover and mutation to create new melodies based on existing ones. Fitness evaluation and selection of superior individuals for breeding the next generation contribute to satisfactory results in generating improvised jazz solos. However, challenges related to mapping chords and scale-to-chord progressions are acknowledged.

The paper “Melody Transcription Via Generative Pre-Training" [4] addresses the conversion of music audio into digital information focusing on musical elements. The system automatically transcribes music recordings into non-overlapping note sequences using music information retrieval techniques. The paper “An Emotional Symbolic Music Generation System Based on LSTM Network" [5] presents a biaxial LSTM network for generating polyphonic music based on human emotions categorized using Russell’s Valence-Arousal emotional space and generates music to match desired emotions.

The authors of “Generating Music with a Self-Correcting Non-Chronological Autoregressive Model" [6] investigate the use of image-based and frequency-based techniques for music generation allowing user control over note-level edits, including correcting off-beat notes. As they utilized U-Net Architecture, our system also takes into consideration the feedback from the users.

"Differential Music: Automated Music Generation Using LSTM Networks with Representation Based on Melodic and Harmonic Intervals" [7] focuses on generating music based on melodic movement and overall musical structure rather than specific pitch or frequencies. The encoding utilizes a one-hot technique with equal-time quantization slices. LSTM networks are employed for training while addressing the challenge of repetitive patterns in generated music.

To mitigate the issue of repetitive patterns in music generated by RNNs, the authors of the paper “Sequence Tutor: Conservative Fine-Tuning of Sequence Generation Models with KL-control" [8] introduce the use of Reinforcement Learning (RL). The use of Fine-tuning RNN with RL was done to adhere to music theory rules. This invites intensive training to address music theories which can be computationally expensive.

The preceding studies showcase sophisticated architectures for melody generation; however, they lack quantifiable assessments of the generated music, relying mostly on subjective evaluation. Furthermore, the produced music does not adhere strictly to any particular scale, and the concept of diverse rich modes is conspicuously absent. The deficiency in mode-specific music generation underscore the need for the authors to address the existing research gap.

3. Methodology

The overall approach to generating music in the presence of various contexts is presented in this section. Entire stages from the data preparation, mode classification to finally generating sequence of relevant notes, chords and tempo are performed accordingly.

3.1 System Architecture

The proposed system has an algorithm differentiating music datasets into particular modes and their root notes implementing concepts from music theory. The proposed system’s workflow can be represented by the block diagram of Figure 1.

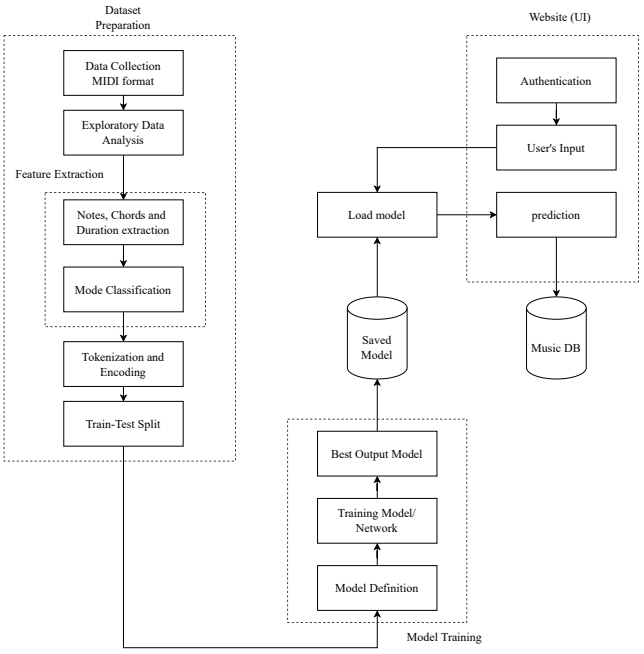

Figure 1: Block Diagram of System Architecture

3.2 Mode Classification

To determine the mode of any given song we used the Mode-Classification pipeline based on a comparison of the notes of the song to the notes in any given mode. The pipeline

employs a series of steps such as key and scale analysis, note transposition, counting of notes, and Hamming distance calculation. The MIDI files store numeric values for each note of music and then store them in binary form. Music21 library is used to recognize the notes based on their MIDI numbers and extract them. Each note map to the natural frequency of sound. Since the midi files assign unique numbers to each of the notes, it's necessary to do the conversion. The conversion of the natural frequency into MIDI number is done with the following formula:

$$\text{MIDI number} = 12 \times \log_2\left(\frac{\text{frequency}}{440}\right) + 69 \quad (1)$$

For obtaining the pitch class from the MIDI number, modulo 12 was applied to the MIDI number. The corresponding output was the respective pitch class which helped in determining the root note of a MIDI song.

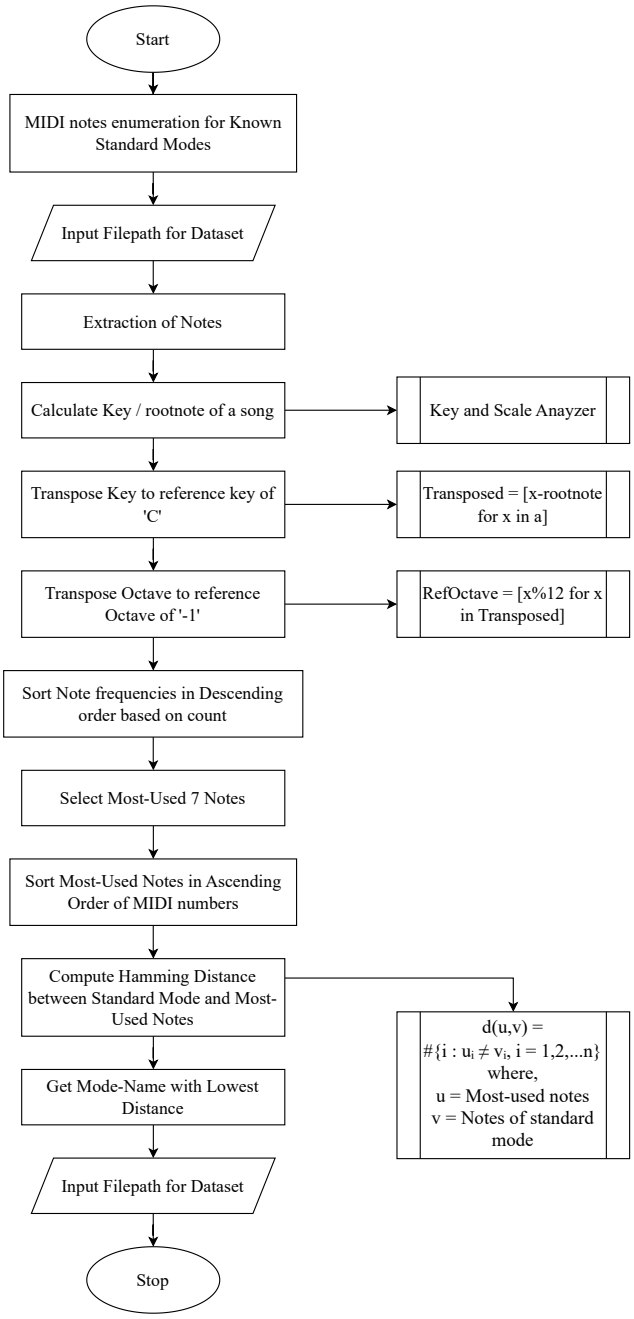

Figure 2: Flowchart for Classification of Modes

Previously, a mode list was created that contains the names of different modes and their corresponding set of notes. This generated list was used for comparison with the notes extracted from the audio file.

3.3 Tokenization and Encoding

Each musical file needs conversion into a proper format that represents each building block of any musical piece which was done in the tokenization process. The extracted note was converted into the string form which defined the individual token. In the case of a chord, each note within the piece was dealt with individually to convert into the string and tokenize respectively. Moreover, the addition of the "⟨SOC⟩" and "⟨EOC⟩" tokens were done at the start and the end in order to wrap the notes present in a chord. In the case of tempo, each tempo obtained was rounded off to get the fixed set of tokens defined by token representation.

The token representation of music was converted into label encoding which was mapped to its corresponding label (number) from the mapping dictionary. Generation of the mapping dictionary was done by appending all the notes in 9 (0 to 8) octaves. The example elements of the dictionary were 'C0': 1, 'C#': 2, and so on. Adding on, a similar case was repeated with the encoding of tempo as well. For instance, the example elements of the dictionary were '30': 1, '50':2, and so on. By knowing the key as a token from the tokenized sequence, we mapped to the corresponding value of dictionary, which in turn provides the encoded sequence. Thus, the encoded sequence after splitting into train and test sets, was suitable to feed into our actual model.

3.4 Generative Network

The core of the generative block [9] is based on two different pipelines used to process "Notes and Chords" and "Tempo" separately. The model architecture consisted of two embedding layers, five LSTM layers, and four fully connected layers. It was trained on a dataset of MIDI files to generate music.

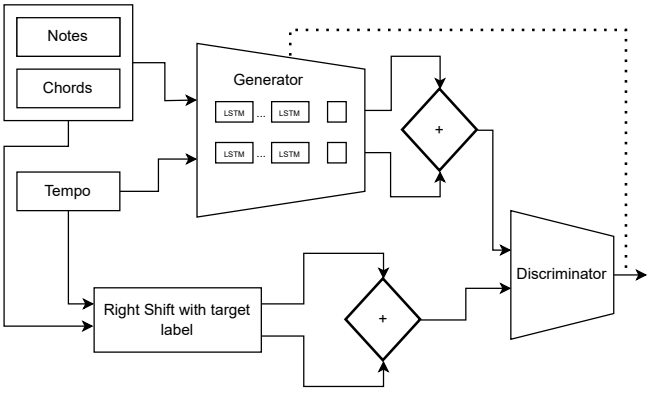

Figure 3: GAN based Model Architecture

The input to the model was a sequence of label-encoded vectors representing musical notes and tempo. Two embedding layers were used to transform the input sequences into fixed-length vector representations. The first layer

handled notes and chords, while the second layer processed tempo information. The input shape was (None, timesteps), where 'None" represented the batch size and "timesteps" was set to 60. The output shape of each embedding layer was (None, timesteps, 100), with 100 representing the embedding dimension per token.

The output of the first embedding layer passed through three LSTM layers with 512 hidden units each. The output of the second embedding layer went through two LSTM layers, also with 512 hidden units but with 512 timesteps. The LSTM layers learned temporal dependencies and captured the long-term structure of the music.

The output of the final LSTM layer for note and duration generation was flattened and passed through two fully connected layers with 256 and 110 neurons, respectively. These layers mapped the LSTM features to a context vector of dimension (256, 1). The last layer used a SoftMax activation function to produce a probability distribution over the 110 possible musical tokens. Similarly, the output of the LSTM layer for tempo generation was flattened and passed through two fully connected dense layers with 128 and 9 neurons, respectively.

The second fully connected layer applied a linear transformation followed by the SoftMax activation function. The SoftMax function normalized the outputs to ensure a sum up to one, representing a probability distribution. The distribution covered 110 distinct tokens for note generation and 9 tokens for tempo generation. Among the 110 tokens, 108 represented notes across 0 to 8 octaves, while the remaining 2 tokens denoted "Start of Chord" and "End of Chord" ("⟨SOC⟩" and "⟨EOC⟩") respectively.

3.5 Discriminative Network

The discriminative block enabled to examine the quality of generated music sequences from the above generative block. Both the model generated and the original piece of music is provided to the discriminator as an input parameter.

The newly generated token is combined with the given input in the case of both Chords and Tempo. After clipping the first indexed token which is done by right shifting the input, we got the fixed-length generated sequence of music. Then the separate entity of "Notes and Chords" and "Tempo" were combined to achieve the single latent representation of model-generated music.

In hindsight, the original input sequence is right-shifted along with the label of the classification problem. Thus, the obtained tokens are aggregated to act as a real musical sequence to the discriminator.

Discriminator block in particular contains a sequence of Embedding, LSTM, and Dense layers to facilitate the recognition of real music from the model-generated one. The loss function used to backpropagate the training of the neural network is given by:

$$L(D) = \max\left[\log(D(x)) + \log(1 - D(G(z)))\right] \quad (2)$$

4. Experiments

Here, we provide the details related to dataset collection, implementation details, hyperparameters tuning and results of our experiments.

4.1 Datasets

Numerous music datasets in the MIDI format were collected from different sources on the Kaggle website. Among them, only the datasets having piano as the main instrument was selected. The piano songs varying from different artists such as Bach to Beethoven, Chopin to Liszt were chosen. About 300 songs were collected that were classical pieces of music from the different periodic eras by well-known musicians from different websites like 'freemidi'. Some modern songs were available on different websites in a piano format which was obtained for varied datasets. Likewise, a few midi files were self-composed by our group in order to make our datasets diverse, so as to include songs in even the rare modes. In total, about 1800 MIDI songs were collected for the purpose of music generation.

Implementing the algorithm for classification of modes of the MIDI songs, the wide range of modes were achieved. The diverse modes of the music can be represented by the Figure 4.

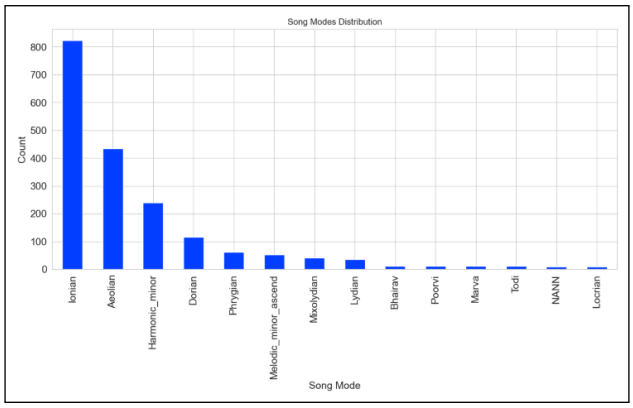

Figure 4: Graph for distribution of data based on Modes of the songs

The majority of the songs taken in our dataset were found to be in Ionian mode. The presence of songs of rare modes like Eastern modes had a low number of songs belonging to them. The representation of both the root notes for each mode is shown in the Figure 5.

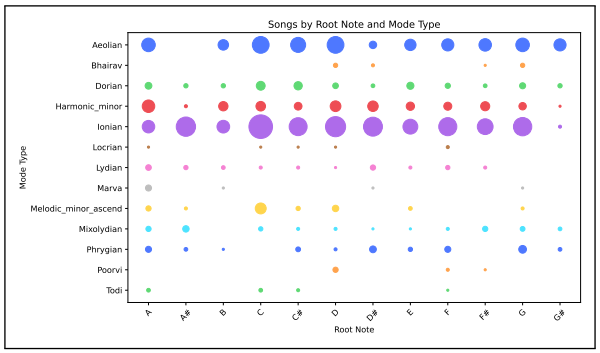

Figure 5: Resulting scatterplot representing the frequency of modes of songs and their root notes

4.2 Implementation Details

For the classification of datasets into modes, the file path was received and the musical information was parsed using the 'Music21' python Library. Before that, the implementation of a mode list was done where each of the numbers is enumerated. The enumeration was done for seven fundamental modes. Likewise, it was done for four Eastern Classical modes (Bhairav, Poorvi, Marva, Todi) and two variations of the fundamental seven modes (Harmonic minor, Melodic Minor).

After being parsed, the percussive aspects of the musical information were discarded from the respective tracks and the melodic elements such as notes and the chords along with their duration were obtained. The Key or root note of the music piece was obtained followed by the transposition of keys and octaves.

For the transposition of the notes in the given key into the reference of the key of C, each pitch was represented in numerical form using the pitch number. Modulo 12 was applied to transpose notes across octaves to the reference octave of -1, corresponding to MIDI numbers ranging from 0 to 11. The frequencies along with their counts were then sorted by implementing a sorting function based on frequency count. Subsequently, the seven most frequently used notes were extracted from the sorted list. The computation of the Hamming distance between the most used notes and modes was performed using the Scikit-learn library.

Say for a given particular piece of music, the seven most used notes in reference to the key of C were [C, D, D#, F, G, G#, A#]. Since these notes were enumerated, starting from 0, the most used notes list was actually represented as: Mostused_notes = [0, 2, 3, 5, 7, 8, 10]

Hamming distance between the Mostused_notes and each of the seven modes was calculated and the following values were obtained.

Table 1: Modes, Note Enumeration, and Hamming Distance

Mode	Note Enumeration	Hamming Distance
Ionian	[0, 2, 4, 5, 7, 9, 11]	3
Dorian	[0, 2, 3, 5, 7, 9, 10]	1
Phrygian	[0, 1, 3, 5, 7, 8, 10]	1
Lydian	[0, 2, 4, 6, 7, 9, 11]	4
Mixolydian	[0, 2, 4, 5, 7, 9, 10]	2
Aeolian	[0, 2, 3, 5, 7, 8, 10]	0
Harmonic minor	[0, 2, 3, 5, 7, 8, 11]	1
Melodic minor ascend	[0, 2, 3, 5, 7, 9, 11]	2
Locrian	[0, 1, 3, 5, 6, 8, 10]	2
Bhairav	[0, 1, 4, 5, 7, 8, 11]	3
Poorvi	[0, 1, 4, 6, 7, 8, 11]	1
Marva	[0, 1, 4, 6, 7, 9, 11]	5
Todi	[0, 2, 3, 6, 7, 8, 11]	2

Thus, the lowest distance is between 'Mostused_notes' and Aeolian, which is 0. So, it can be concluded that the song selected has the most similarity with the Aeolian mode. This concludes that the given piece of music is under the Aeolian mode.

The model was trained on a dataset of MIDI files containing classical piano music. Categorical cross-entropy was used as the loss function and Adam optimizer was used to optimize the parameters. The training process involved iteratively feeding the input sequences into the model and adjusting the weights of the different layers to minimize the loss function. The batch size of 128 was used to feed the training data in batches to the model. The number of epochs was set to 50, which meant that the model went through the entire dataset 50 times during training.

Dropout layers with a rate of 0.3 were added after each LSTM layer during training to prevent overfitting. However, during music generation inference, dropout was not used to ensure the generation of the most likely sequence of notes given the input seed. The loss and accuracy during training were monitored using a validation set. The training process stopped when the validation accuracy stopped improving to prevent overfitting.

4.3 Hyperparameters

Tuning the hyperparameters plays an important role in determining the quality and diversity of the generated music. The choice of hyperparameters can have a significant impact on the model's ability to capture the complex patterns and structure of music. The following hyperparameters were used for training purposes to control the overall behavior of the model.

4.3.1 Learning rate

The learning rate determines the step size at which the optimizer updates the model parameters during training. A smaller learning rate can lead to slower convergence, while a larger learning rate can cause the model to overshoot the optimal solution. In the case of music generation, a learning rate of 0.001 was found to be optimal.

4.3.2 Batch size

The batch size determines the number of training examples used in each iteration of the training process. A larger batch size can lead to faster convergence, but can also require more memory and processing power. In the case of music generation, a batch size of 128 was used.

4.3.3 Number of epochs

The number of epochs determines the number of times the model is trained on the entire dataset. A larger number of epochs can lead to better performance, but can also increase the risk of overfitting. In the case of music generation, a total of 50 epochs were used.

4.3.4 Dropout rate

The dropout rate determines the probability of dropping out a neuron during training. Dropout is a regularization technique that can help prevent overfitting by randomly dropping out some of the neurons in the model. In the case of music generation using LSTM, a dropout rate of 0.3 was used after each LSTM layer.

4.4 Results

4.4.1 Loss Curve

For different modes, different models were trained with the songs matching the modes. As there was low number of changes in the tempo, only the sequence of notes and chords were considered for appropriate calculation of loss and accuracy.

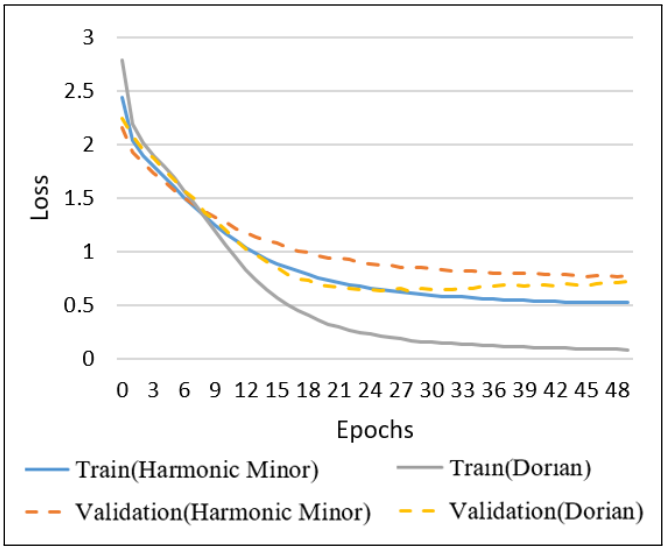

Figure 6: Loss Curve for Harmonic Minor and Dorian Mode

The curves in Figure 6 illustrate the loss of training and validation data for only the notes and chords with respect to different modes over the number of epochs. The difference in the predicted and actual output of the models during the training phase can be visualized from the above diagrams. The sparse categorical cross entropy losses calculated for each training example are summed up to get the final loss of the epoch that is plotted in the graph.

By looking at the curves of Figure 6, we can infer that the loss has gradually decreased over the number of epochs moving from left towards right (0 to n epoch). However, the loss was stabilized after around 32 epochs in the case of the second graph. Also, the loss for the training data of both Phrygian and Aeolian is slightly lower compared to validation data which is good enough to make our models in a low variance state. Moreover, we can also conclude that the models have a low bias for its loss lying in the range of 0 to 1.

4.4.2 Accuracy Curve

The accuracy curves in Figure 7 depict the performance of our overall machine learning model over the various musical modes. The individual accuracies calculated over the training examples are added to return the actual accuracy for the given epoch. The accuracy was seen to be increasing exponentially with the increment of epoch till the saturation level (n epoch). Since the gap training and validation accuracy is fairly narrower, it can be concluded that our models are in a low variance state. It is regarded that a music generation system can predict consecutive notes with less error provided the accuracy is high.

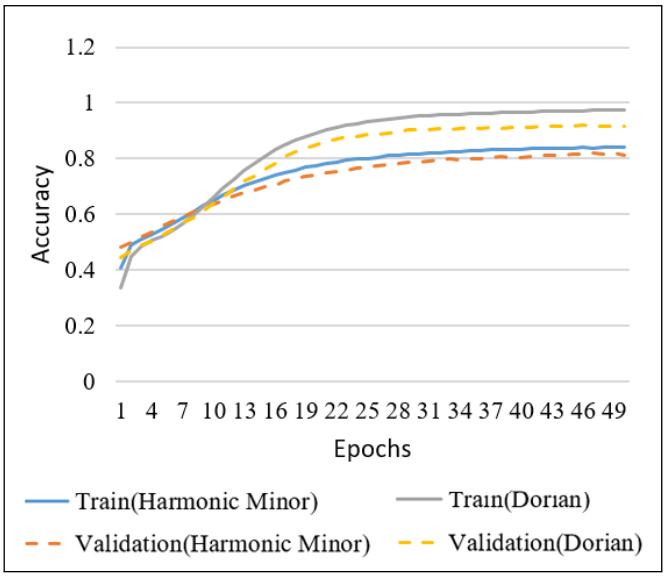

Figure 7: Accuracy Curve for Harmonic Minor and Dorian Mode

Thus, from the loss and accuracy curves, it can be inferred that our machine learning models make fewer errors in minimal input when interfaced with numerous input samples.

4.4.3 Confusion Matrix

The confusion matrix in Figure 8 is created for the Ionian mode to visualize the predicted notes by the model given the actual notes from the validation dataset. It enables the model to assess its performance without confusion in choosing from the range of classes (notes). The numbers in the diagonal of the matrix symbolize the count of True Positives (TP) predicted in the Ionian mode. Also, the summation of the non-diagonal elements gives the count of wrongly classified notes by the model. To enhance clarity, the color intensity over the confusion matrix depicts the count in each cell [i, j], where i represents the row number and j represents the column number. It is observed that the required intensity in the diagonal, as in a typical confusion matrix, is missing. However, the intensity over non-diagonal elements is quite satisfactory compared to the standard confusion matrix.

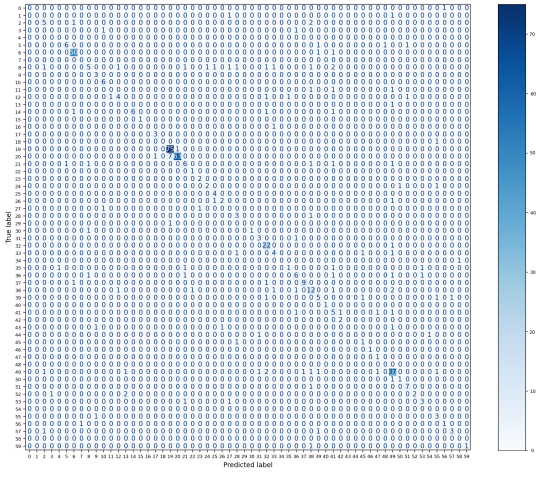

Figure 8: Confusion Matrix for Ionian Mode

Since Ionian had the largest number of songs in our dataset, Ionian mode was chosen for computing F1 scores. Similar scores could be achieved for each mode consequently. For further evaluation, an online survey was conducted by allowing others to rate the music pieces produced by our model for each mode.

Table 2: Classification Metrics

Metric	Precision	Recall	F1-score	Support
Macro average	0.500262	0.516335	0.490216	500
Weighted average	0.759995	0.758	0.74893	500

Accuracy : 0.758

4.4.4 Crowd Evaluation

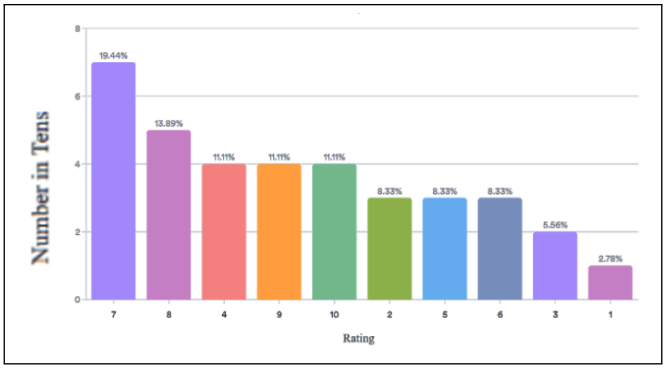

Figure 9: Graph to show the rating of songs in an average of all modes

The graph in Figure 9 demonstrates the average rating for all modes of the demo songs averaged in total. About 19 percent of the surveyed individuals rated the songs 7 on average. About 2 percent of the individuals rated '1' for the songs produced by our model. A range of ratings can be seen from the feedback from the audience, but it can be considered positive feedback from the overall perspective.

The individuals who participated were also asked about their experience in the field of music. Different individuals with varying experiences were part of the survey. The graph of Figure 10 demonstrates that those with less or higher experience along with the ones having no clue took part in the survey as the audience.

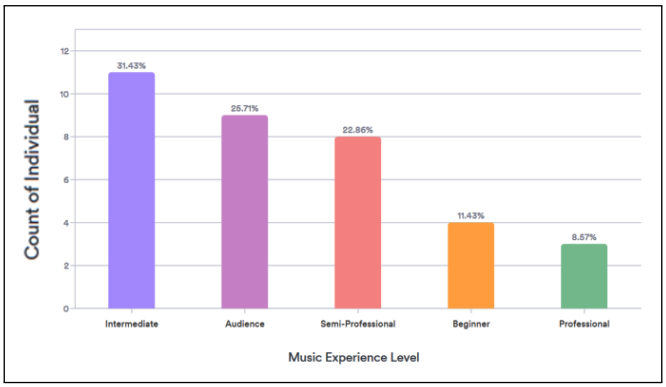

Figure 10: Graph of Music Experience of the Audience involved in the survey

5. Conclusion

The design of an autonomous music composition system is a challenging task in itself. Knowing which set of notes to play with new patterns can be computationally expensive. Thus, music generation with the concept of the composition of new melodies is a creative task. Due to the lack of modal datasets, there is a need to create our own dataset. The modal dataset is to be created by applying the mode-classification algorithm to the collected dataset. The newly created dataset is then used for the training of the songs together that have similar modes.

Thus, the system developed in this project successfully and accurately classifies the modes of the dataset. The choice of selecting mode, root-note, tempo, and time signature is provided to the user. When the user selects one's desired choice, the trained model successfully predicts new notes/chords based on the input provided. The time taken by our system to generate music varies depending on the complexity of the composition and the computational resources available, which is usually a couple of seconds.

To summarize, the feedback from the surveyed individuals yielded a satisfying and positive response. The respondents provided ratings for the music anonymously, with nearly 8% identifying themselves as professionals in the field of music. Their assessments served as the subjective measurement of the produced music. Moving forward, the inclusion of the additional musical elements with a better model for the neural network architecture would further enhance this project in the future.

References

[1] Curtis Hawthorne, Ian Simon, Adam Robert, Neil Zeghidour, Jordan Gardner, Ethan Manilow, and Jesse Engel. Multi-instrument music synthesis with spectrogram diffusion. In *International Society For Music Information Retrieval Conference*, 2022.

[2] Rui Guo. An interactive music infilling interface for pop music composition. *ArXiv*, abs/2203.12736, 2022.

[3] John D Biles. Genjam: A genetic algorithm for generating jazz solos. In *International Conference on Mathematics and Computing*, Kaohsung, Taiwan, 1994.

[4] Chris Donahue, John Thickstun, and Percy Liang. Melody transcription via generative pre-training. In *International Society For Music Information Retrieval Conference*, Bengaluru, 2022.

[5] Kun Zhao, Siyuan Li, Chen Junanjuan, Hui Wang, and Jun Wang. An emotional symbolic music generation system based on lstm network. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 2039–2043, March 2019.

[6] Wei Chi, Prashant Kumar, Sudeep Yaddanapudi, Ram Suresh, and Umut Isik. Generating music with a self-correcting non-chronological autoregressive model. *ArXiv*, abs/2008.08927, 2020. 18 August 2020.

[7] Hesam Rafraf. Differential music: Automated music generation using lstm networks with representation based on melodic and harmonic intervals. *ArXiv*, abs/2108.10449, 2021.

[8] Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, Jose Miguel Hernandez-Lobato, Ryan E Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *International Conference on Machine Learning*, New York, 2016.

[9] Ian J. Goodfellow et al. Generative adversarial nets. *Communications of the ACM*, 63(11):139–144, 2020.