Generating Frank Ocean Music Using Deep Learning

Philani Mpofu
School of Computer Science and Applied Mathematics
The University of the Witwatersrand
Johannesburg, South Africa
Email: 1848751@students.wits.ac.za

Ritesh Ajoodha
School of Computer Science and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
Email: ritesh.ajoodha@wits.ac.za

***Abstract*—The algorithmic composition of music that abides by musical rules and structure is a problem that computer scientists have spent many years attempting to solve. In their pursuits to solving this problem, researchers have used many methods that range from Markov chains and genetic algorithms to deep learning and Bayesian analysis. This problem domain is enticing because, essentially, it seeks to use algorithms and computer structures to model an inherently human task. To date, these attempts generated models that aimed to algorithmically compose music of a specific genre or music influenced by several genres. In this case, the research seeks to build on these previous attempts by contributing a process that uses deep learning and natural language processing techniques to algorithmically compose music similar to that of a particular artist - Frank Ocean.**

***Index Terms*—Algorithmic Music Composition, Natural Language Processing, Deep Learning, Frank Ocean**

I. Introduction

Frank Ocean is arguably one of the most acclaimed artists of his generation. A master of confessional songwriting, Frank Ocean earned cult-icon status with his enigmatic persona and idiosyncratic approach to R&B [1]. Influenced by electro-funk, pop-soul and even psychedelic rock, Frank Ocean continually defies the rigid conventions of R&B, and his music speaks for itself. At the 2013 Grammy Awards, his debut studio album *channel Orange* won Best Urban Contemporary Album and was a contender for Album of the Year. His second album *Blonde* has not left the US Billboard Top 200 Chart since debuting at number one in 2016. Despite the success of his sophomore album, Frank Ocean has not released a full-length album since 2016, and fans are yearning to reconnect with him. At this point, it seems as if deep learning is the only way to bring him back from his hiatus and spur him out of his reclusivity.

There have been many attempts to generate music using deep learning techniques. However, these attempts merely aimed to compose instrumental music of a single genre or instrumental music influenced by multiple genres. The most common approach is to train neural network architectures to learn time-dependent patterns and inherent musical structure in training data to compose music similar to that used for training. However, there have been no attempts to use this approach to reverse engineer and replicate songs made by a specific artist. This report aims to build on these previous attempts by using deep learning neural network architectures and natural language processing techniques to generate music in the likeness of a particular artist; in this case, that artist is Frank Ocean. Essentially, the generative model will generate musical output that replicates Frank Ocean's singing voice, his songwriting style, as well as the way he structures his songs.

Section II is a review of previous attempts made by other researchers to solve similar problems to give the reader a better understanding of how the research report relates to the broader context of algorithmic music generation. Section III presents the research methodology, such as the format used to store training data and the deep learning models used to generate output. Section IV discusses the metrics used to measure the performance of the generative process and the final results.

II. Related Work

Over the years, many researchers have attempted to use deep learning techniques to algorithmically generate instrumental music of a single genre or instrumental music influenced by multiple genres. Most research papers solving this problem follow a similar process: train a neural network architecture to learn time-dependent patterns and the inherent musical structure in training data to generate a predictive model that composes music akin to that used during training. This section consists of a review of all the different attempts at solving this problem by exploring the several formats that researchers used to store training data, how they extracted features from the training data, how they implemented and trained deep learning models to generate likeable music, and how they evaluated the performance of these deep learning models.

A. Training Data

An important aspect to consider when implementing any deep learning algorithm is the format that will store the training data. For the algorithmic generation of music, songs make up the training data, and some audio format needs to represent and store this data. A common practice is to use MIDI files to store all of this music. A MIDI file is an audio file that contains an almost symbolic representation of music, and one could even think of it as a form of digital sheet music. Thus, this format is very appropriate for the algorithmic composition of music. This is because any training on these files can be done with a more music theory based approach since the implemented structures will have access to features such as notes, chords, and the progression of musical events. [2] used a dataset made up of thousands of MIDI files to train a recurrent neural network (RNN) to remember musical notes and use those notes to predict the next notes in a musical sequence. There are other formats available to store songs for training. However, these formats are very limited in their capabilities. [3] used WAV files to store their training data, and they had no access to music theory based features. [4] used a dataset consisting of 4235 lead sheets and 120 jazz solo transcriptions as training data. Since an analogue format stored the training data, features, such as measures of music, needed to be derived manually.

Another aspect to consider in the algorithmic generation of music is the genre of the training data. The genre of the training data loosely determines the genre of the generated music. Take, for example, a training dataset that consists solely of pop music. Any neural network architecture that is trained on this dataset will compose music that has a lot of the traits of pop music, such as repeated choruses and fast tempos. [5] used a dataset of 150 classical piano pieces (22 of them were from Bach's fugues) to train an RNN. The RNN then composed classical piano pieces that were influenced by Bach. [6] used a dataset made up of rock and jazz music to train an RNN. The RNN then composed pieces that were influenced by rock and jazz. In some papers, there was no consideration given to the genre of the training data. In this case, the computer-generated music was an aggregation of the genres of the songs in the training dataset. [7] used a dataset of 30 000 songs to train an RNN. This RNN then composed likeable music that was unclassifiable under a single genre.

Generating music in the likeness of Frank Ocean would build on the practices described above. The methods above aimed to compose music categorized under a specific genre or music unclassifiable under one genre using digital or analogue formats for storing training data. On the contrary, the research problem intends to use machine learning algorithms to generate music similar to Frank Ocean's. Any neural network architecture implemented to compose Frank Ocean music would specifically use Frank Ocean's entire discography as training data and not merely music from any particular genre. Thus, it would be appropriate to store all of his music in a digital format, such as a MIDI file, because any architecture trained on this dataset would have access to music-theory based features.

B. Features

As highlighted in the Training Data section, there are numerous ways to store songs as training data, but there are only a few ways to extract features from the training data. Only features accessible or available in the format used to store the training data are extractable. Take, for example, MIDI files. MIDI files store features associated with musical performance, such as time signatures, melodies, tempos and key signatures, to name a few. [6] extracted notes, chords, pitches and duration as features from MIDI files in the training dataset. In addition to all of those MIDI events, [2] also extracted message types and velocities as features. [7] extracted monophonic melodies as features, and [5] extracted polyphonic melodies as features. Other formats had certain limitations, and thus not as many features could be extracted from them. Since the training data was stored in an analogue format, [4] could only extract single measures of music as features, and [3] could not extract any features from the WAV files in their training dataset.

As mentioned above, Frank Ocean's entire discography would need to be stored and represented in a digital format, such as a MIDI file, because such formats contain the relevant music-theory based features necessary for learning the characteristics of songs unique to Frank Ocean. In addition, the values within those features would be specific to Frank Ocean, and any architecture that extracts those features can better emulate his style of music.

C. Methodology and Evaluation

Neural networks are proven to be highly efficient in solving problems in a wide range of topics, namely classification, clustering, and, most importantly, pattern recognition. When trained on the appropriate dataset and implemented as generative models, neural networks can generate output similar to human-made art. There have been attempts to use neural networks to extract and generate complex sequential arrangements of musical segments. Nonetheless, they were insufficient in capturing the time-dependent musical structure [8]. Several papers believed that a possible solution to this problem could be an RNN because it can recall time-dependent information, such as features from training. However, [9] concluded that this network architecture performs poorly in music generation, citing the vanishing gradient problem and other unwanted behaviour. [10] developed The Long Short-Term Memory (LSTM) to overcome this limitation.

LSTMs are a type of RNN that use gradient descent for efficient learning. Similarly to RNNs, LSTMs can recognize and encode patterns over time-dependent intervals. Furthermore, LSTMs overcome the vanishing gradient problem observed in RNNs. Thus, LTSMs are a practical model for music generation [11]. [5] implemented an LSTM model that can recall past details from the training data and learn the fundamentals of those details to compose a well-organized polyphonic output that abides by musical rules and structure. In addition, [5] trained their model on songs of gradually increasing degrees of difficulty because the high variance observed in music segments would disorient a recently initialized model. Using the log-likelihood [5] concluded that the LSTM model is appropriate for music generation. They recommended that incorporating a reinforcement learning paradigm would allow for effective modelling of the underlying musical structure and increased range in potential musical expression.

Moreover, LSTMs can produce even better results when implemented in conjunction with other machine learning paradigms. [7] trained an LSTM on a set of music data to generate the following note in a musical sequence. Following the training stage, the LSTM is optimized using reinforcement learning techniques. In this case, the reward function is based on music theory and the output of another trained LSTM. Using a Turing Test and their model's adherence to music theory, [7] concluded that implementing a combination of unsupervised and reinforcement learning structures can generate a better model and compose higher quality music than solely implementing an LSTM network. [7] also concluded that their approach reduces the instances of unwanted behaviour of the LSTM, such as generating sequences with too many repeated notes or generating sequences with no inherent structure.

Since LSTMs produce good results when implemented in conjunction with other machine learning paradigms, one would assume that the same holds for when LSTMs are implemented in conjunction with other neural network architectures to generate music. However, certain combinations of these architectures are not appropriate. In their unit selection-based approach, [4] used an LSTM with a deep structured semantic model (DSSM) to generate monophonic melodies. Given a unit from the training data, the predictive model uses both the DSSM and the LSTM to generate the next unit. In addition, this model ranks units according to a semantic relevance score (calculated by the DSSM) and a concatenation cost (calculated by the LSTM). Using the mean rank, accuracy, and Turing Test as metrics, [4] concluded that this approach is too constrained and that adding some supervision on the note-level may be a necessary measure to implement to create likeable music. [2] implemented an LSTM-GAN hybrid architecture to generate music. A generative adversarial network (GAN) is another architecture capable of learning patterns in training data. The approach detailed in their report involved training two LSTM models with conflicting roles: one a generator and the other a discriminator. The generator produced some random noise as output, and the discriminator had to differentiate between the sample music file and the sample file with generated noise. The generator would then adjust its weights according to the error observed by the discriminator between the original music sample and the sample with generated noise. This process continued until the discriminator was no longer able to distinguish between the original music sample and the generated sample. Using a Turing Test, [2] concluded that an LSTM-GAN hybrid approach performs very poorly in music generation.

Although some of the methods mentioned above are applicable for algorithmically generating music similar to Frank Ocean's, they are limited because a few components, such as song lyrics and Frank Ocean's singing voice, have not been addressed. The methods mentioned above are only applicable for generating instrumental music in Frank's likeness and not entire songs with his voice singing over the instrumentals. The generative model must learn the inherent structure of Frank Ocean's song lyrics and then generate song lyrics that abide by this structure. Furthermore, the generative model must also learn the inherent composition in Frank Ocean's voice to replicate his singing voice, inflexions and pronunciations, thus keeping Frank's essence and artistic style in the composed output.

III. Methodology

This section introduces the procedure for addressing the research problem and achieving the research aims and objectives. The subsections in this section are split into two subsubsections: text and audio. The text refers to the textual input and output of the generative model, that is, song titles and song lyrics. Audio refers to the audial input and output of the generative model, that is, instrumentals and Frank Ocean's singing voice.

The textual and audial training data is extracted from various sources and converted to an appropriate format. The Audial data is stored in a collection of WAV files, and the textual data is stored in a single text file. The text data is used as input for an RNN to generate textual output, and the audial data is used as input for an LSTM to generate audial output. The final output is then evaluated using music theory-based metrics.

A more detailed explanation of the research methodology follows in subsequent subsections. Subsection III-A describes the various formats used to store textual and audial training data as well as the ethical implications of these chosen formats. Subsection III-B presents the language modelling techniques to generate textual output and the deep learning model implemented to generate audial output. Finally, Subsection III-C describes the overall weaknesses of the chosen methodology.

A. Training Data

The training data will consist of the audial and textual components of Frank Ocean's mixtape *Nostalgia, ULTRA* (2011) and his two studio albums *channel Orange* (2012) and *Blonde* (2016) web-scraped from multiple online sources.

These three albums are his most popular releases, and collectively they make up over 80% of his entire discography. Any generative model trained from this dataset will generate output that encompasses the core components of Frank Ocean's artistic style.

Textual Training Data

Frank Ocean's song lyrics are web-scraped from the Genius lyrics library using the lyricsgenius python package. This package acts as a wrapper around the Genius API and provides a simple interface that allows for the straightforward extraction of textual data, such as song titles and song lyrics, into a text file. This text file is the textual training data. The only preprocessing performed on this training data is a string to numeric mapping. Before training, all the words in the dataset are mapped to an integer value, giving them a unique numeric representation. The motivation behind using a numeric representation for all words in the dataset instead of the original string is because computers can only recognize numbers and not strings of text. Thus, every word is converted to this format before being passed on to language model.

Other lyrics libraries, such as AZ Lyrics and LyricsFreak, are available for textual data extraction. However, these libraries are not as comprehensive as the Genius library. Genius is the world's most established lyrics library, and it undoubtedly contains all of the relevant Frank Ocean textual data necessary for sufficient training, whereas other lyrics libraries may not. In addition, the lyricsgenius package is only compatible with the Genius API. Using this package allows for a more automated approach to the acquisition of the training data. Instead of having someone manually copy all of this metadata from the Genius website and pasting it onto a text file, one can merely write a function using Python code to save all of this information into a text file.

Audial Training Data

Frank Ocean's songs are web-scraped from various sources and saved in the training dataset in an MP3 format. These songs are converted from an MP3 format to a WAV format using the publicly available digital audio workstation, Audacity. WAV files store the training data because WAV is the only format compatible with the pre-trained LSTM architecture highlighted in subsection III-B of this report. Although this is already the case, other formats, such as the MP3 format, are insufficient for training. MP3s are inherently a lossy format and could be missing valuable data points as a result of compression.

Frank Ocean's two studio albums *channel Orange* and *Blonde* are not in the public domain, as they are both owned by a record label. Neither Def Jam (which owns *channel Orange*) nor Boys Don't Cry (which owns *Blonde*) have approved their respective albums to be used as training data for this research project. Abiding by copyright laws, only 30-second clips of these songs will be used as training data. His mixtape, *Nostalgia, ULTRA*, was publically released and can be used in its entirety for training.

B. Deep Learning Models

Text Generation

New Frank Ocean song titles and song lyrics were generated using a language model. A language model is a model capable of understanding the rules and structure of a given language. Given a sequence of words from the language, the model uses its understanding of the language to predict the following sequence of words. Essentially, language models are conditional probability models of the form:

$$P(\mathbf{W}|\mathbf{T}) = \frac{P(\mathbf{W} \cap \mathbf{T})}{P(\mathbf{T})} \quad (1)$$

where $\mathbf{W}$ is the following sequence and $\mathbf{T}$ is an array of sequences of words before $\mathbf{W}$ (ie: the context of $\mathbf{W}$) [12]. Figure 1 illustrates the deep learning architecture of the language model.

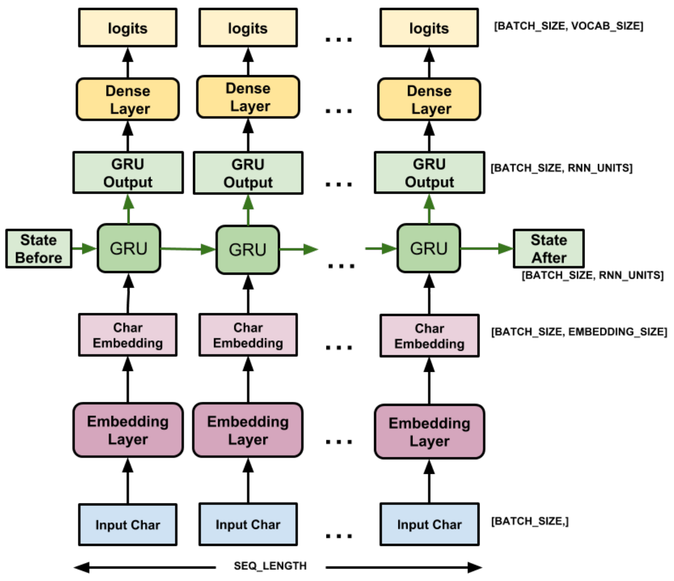

Fig. 1: Illustration of the Language Model

The language model in the research is a three-tiered model built in Python with Keras and Tensorflow. It has an embedding layer for an input layer. An embedding layer is a trainable lookup table that maps each word in the dataset to a vector. The second layer is a GRU layer. Essentially, the GRU layer is an improved version of an RNN. Finally, the output layer is a dense layer. It outputs one logit for every word in the textual training dataset. A logit is the log-likelihood of each word according to the learned language model. To predict the next word, the model looks up the embedding for each word in the dataset, runs the GRU for one timestep, and uses the dense layer to generate logits predicting the log-likelihood of the next word [10]. Figure 2 presents the training loss curve of the language model after 50 epochs. Once the model has completed training, we generate song lyrics for a given song title.

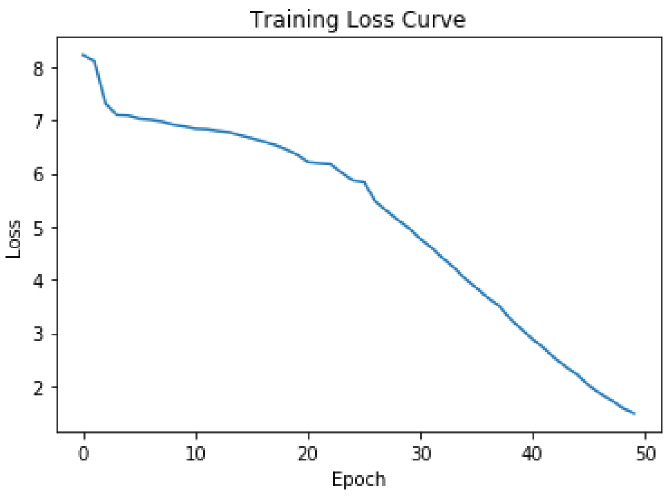

Fig. 2: Training Loss Curve of the Language Model

To generate a song title, we first randomly select a number n between (and including) 1 and 4. This number denotes the number of words in the song title. From here, we randomly select n number of words from the training dataset until we get a pleasant result. Once we have the best sounding song title, we use this title as the start string or context for the language model. The language model generates 120 words from the given start string. Again, we repeat this process until we get a pleasing result for the song lyrics of the start string. Once satisfied with the lyrics, we remove any punctuation or grammatical errors to assist the audio generation model.

Several researchers have concluded that RNNs are not suitable architectures for music generation. However, due to their ability to recall time-dependent data during training, these architectures are more appropriate for text generation. A Bayesian network is another suitable model for text generation. However, it is not implemented in this project because it is computationally more expensive than the RNN.

Audio Generation

The open-source Jukebox AI architecture from OpenAI is used to generate new Frank Ocean music. Jukebox AI is a deep learning neural network that generates music, including rudimentary singing, as raw audio in several genres and artistic styles [13]. OpenAI has already trained the architecture on a dataset of 1.2 million songs. Thus, the Jukebox AI is only used for inference for this project. A Frank Ocean song (paired with a set of lyrics we generated in the previous step with the language model) is fed to the Jukebox AI. The Jukebox AI then uses its understanding of music to gain a machine understanding of the inherent structural relationships found in the song [14]. From this understanding, the Jukebox AI generates a song that should sound like something Frank Ocean would release. Figure 3 illustrates this process.

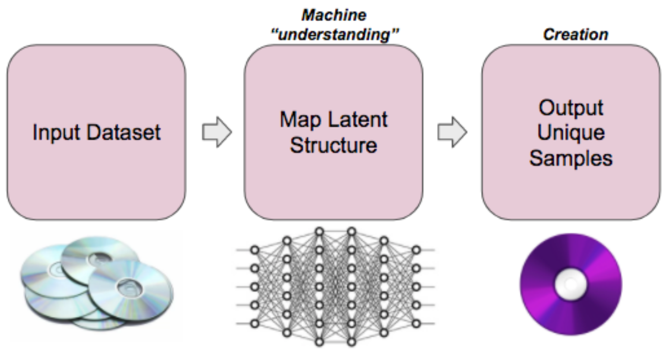

Fig. 3: Illustration of the Jukebox AI Generative Process

The Jukebox AI is built in Python with Keras and Tensorflow. Essentially, the Jukebox AI is a bidirectional LSTM architecture. It consists of two LSTM layers, two dense layers and a dropout at each layer. The first LSTM layer trains on the progression of musical events using forward propagation, and the second trains on the same arrangement using backpropagation. The LSTM architecture can recognize and encode patterns over time-dependent intervals, an essential trait required for music generation. Figure 4 illustrates the architecture of the Jukebox AI. The standard RNN is another architecture capable of generating a time-dependent predictive model. However, it performs poorly in music generation due to the vanishing gradient problem.

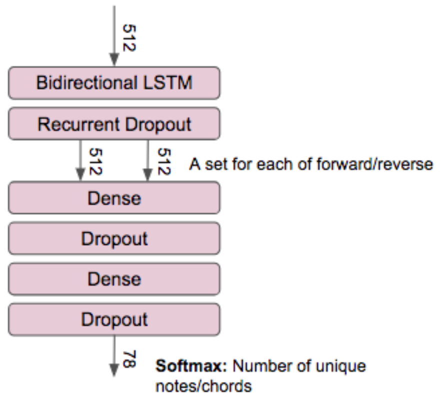
512
Bidirectional LSTM
Recurrent Dropout
512
512
A set for each of forward/reverse
Dense
Dropout
Dense
Dropout
78
Softmax: Number of unique notes/chords

Fig. 4: Illustration of the Jukebox AI Architecture

C. Limitations

The trade-off between the model's complexity and the time required to train the model limited the conclusions of the research. Complicated models take several days to train, while less complicated models produce output with reduced dimensionality. Due to the limited availability of hardware, the predictive models need to be less complex to generate output in a reasonable amount of time. However, this contrasts Frank Ocean's artistic style, which is notorious for being complex and multifaceted. A simpler model dilutes his individuality. In addition, the Jukebox AI is also limited in its capabilities. The songs it has generated have shown local musical coherence, such as following traditional chord patterns and composing impressive solos. However, the AI is not capable of generating music that follows a more global structure, such as repeating choruses.

IV. Evaluation and Results

The nature of the research problem is generative. Thus, a qualitative or subjective metric would be most desirable to measure the performance of the generative models. Therefore, one could conduct a Turing Test where the survey group comprises people already familiar with Frank Ocean's music. This group would listen to the generated output and then determine, based on the sound, lyrics, and song titles, if the music sounds similar to Frank Ocean's. If the majority of the group agrees that the generated music sounds like something Frank Ocean would release, then the generative models performed well. If not, then the generative models performed poorly, and there would need to be an investigation into why that is the case. This approach may seem appealing. However, it is not very conclusive. It is challenging to conduct an experiment in this manner that leads to valid, reliable, and replicable results as it is incredibly difficult to control all of the relevant variables, eliminate bias, and recruit a sufficient number of qualified subjects [15]. Thus, a quantitative or objective metric would be most desirable to measure the performance of the generative models.

We use quantitive metrics to evaluate the model because they give great insight into the properties of the generated output. First, we describe the two datasets. One dataset contains the ground truth dataset, and the other dataset contains the generated output. We then extract a couple of musically meaningful and comprehensive features, pitch and tempo. After extracting these features for both datasets, we compute objective measurements, such as the count, range, mean and standard deviation for each feature. We then compare the differences between the ground truth dataset and the generated output according to these metrics.

A. Pitch-Based Features

Pitch refers to the fundamental frequency measured in hertz. The fundamental frequency is simply the musical pitch of a note perceived as the lowest partial present [16]. The following pitch-based features are used to evaluate the performance of the audial generative model:

1) **Pitch Count**: The number of unique pitches for a given dataset.

2) **Pitch Range**: The difference between the highest and lowest pitch in a given dataset. It measures the spread of extreme pitch values for a given dataset.

3) **Pitch Mean**: The average pitch for a given dataset. It can also be thought of as the expected pitch value for a given dataset.

4) **Pitch Standard Deviation**: A measurement of how dispersed a given dataset is in relation to the pitch mean. A low value implies that the pitch values are clustered around the mean, and a high pitch standard deviation implies that the pitch values are more spread out from the pitch mean.

The table below summarizes the original audial dataset and the generated dataset using the above metrics.

TABLE I: Original Dataset vs Generated Dataset Pitch-Based Features

Dataset	Count	Range	Mean	Standard Deviation
Original	274	386	189	83
Generated	213	374	230	54

An overview of the above table indicates that both datasets are very similar with regards to pitch. Both datasets have almost the same number of unique pitch frequencies, and there is an almost perfect spread of pitch frequencies in both datasets. However, the generated mean pitch frequency is significantly higher than the original mean pitch frequency. In addition, the original pitch standard deviation is much higher than the generated pitch standard deviation. These observations imply that the audial generative model generates higher pitch frequencies than Frank Ocean's original music on average. This relationship is visualised with a pitch class histogram as shown in figure 5.

The two datasets appear to be similarly distributed. We want to quantize this similarity by calculating the statistical distance between the two probability distributions using the Jensen-Shannon Distance (JSD). The JSD measures the similarity between the two probability distributions. It returns a scalar value between 0 and 1. A JSD of 0 indicates that the two distributions are the same, and a JSD of 1 indicates that the two distributions are dissimilar. Equation 2 presents the formula used to calculate the JSD.

$$JSD(P,Q) = \sqrt{\frac{1}{2} \cdot KLD(P,M) + \frac{1}{2} \cdot KLD(Q,M)} \quad (2)$$

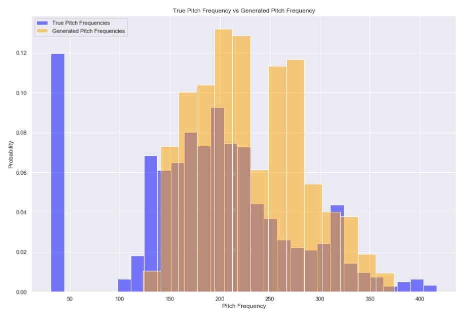

Fig. 5: Pitch Class Histogram of Original and Generated Datasets

In equation 2, KLD refers to the Kullback-Leiber Divergence, which is another measure of statistical distance. P is the pitch class histogram of the original dataset, and Q is the pitch class histogram of the generated dataset. M is another probability distribution equal to half the sum of P and Q. The JSD for P and Q is 0.63. This value implies that the original dataset and the generated dataset are neither similar nor dissimilar. Thus, the model performed well when looking at pitch based features as it generated a new distribution that maintained the relationships found in the original dataset.

B. Tempo-Based Features

Tempo refers to the speed of a piece of music, measured in "Beats per minute" (BPM). The following tempo-based features are used to evaluate the performance of the audial generative model:

1) **Tempo Count**: The number of unique tempos for a given dataset.
2) **Tempo Range**: The difference between the highest and lowest tempo in a given dataset. It measures the spread of extreme tempo values for a given dataset.
3) **Tempo Mean**: The average tempo for a given dataset. It can also be thought of as the expected tempo value for a given dataset.
4) **Tempo Standard Deviation**: A measurement of how dispersed a given dataset is concerning the tempo mean. A low value implies that the tempo values are clustered around the mean, and a high tempo standard deviation implies that the tempo values are more spread out from the tempo mean.

The table below summarizes the original audial dataset and the generated dataset using the above metrics.

An overview of the above table indicates that both datasets are almost alike regarding tempo. Both datasets have the same number of unique tempos, and there is an almost perfect spread of tempos frequencies in both datasets. The generated mean tempo is slightly higher than the original mean tempo. In addition, the original pitch standard deviation is the same as the generated pitch standard deviation. These observations imply that the audial generative model generates faster pieces of music than Frank Ocean's original music on average. This relationship is visualised with a tempo class histogram as shown in figure 6.

TABLE II: Original Dataset vs Generated Dataset Tempo-Based Features

Dataset	**Count**	**Range**	**Mean**	**Standard Deviation**
Original	9	107	101	29
Generated	9	99	106	29

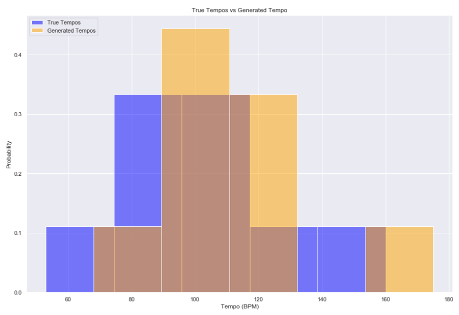

Fig. 6: Tempo Class Histogram of Original and Generated Datasets

The JSD in this case is 0.17. This value implies that the original dataset and the generated dataset are almost identical. Thus, the model performed very well when looking at tempo based features, as it generated a new distribution that maintained the relationships found in the original dataset

Although qualitative evaluation is generally preferable for evaluating generative modelling, it poses a few issues that can invalidate results. Therefore, we chose to use quantitative and musically informed objective metrics to measure the performance of the generative models. In addition, these metrics also allow for the reproducibility of results and provide a means to compare the generated output and the ground truth dataset. Using these metrics, we have shown that the generative model performs very well in generating new Frank Ocean music and still maintains the relationships found in the ground truth dataset.

V. Conclusion

In conclusion, the subject of algorithmically generating music had been thoroughly investigated; however, the research problem had not been adequately solved. Although some of the methods referenced in the related literature are applicable for algorithmically generating music similar to Frank Ocean's, they are insufficient for solving the research problem because elements, such as song names, song lyrics and Frank Ocean's singing voice, have not been adequately discussed. These methods and practices are only applicable for generating instrumental music in Frank's likeness and not entire songs with his voice singing over the instrumentals. The research report contributed a method that utilized deep learning and language modelling to generate audial and textual output similar to Frank Ocean's music. Using objective music theory-based features, we deduced that our modelling process is capable of generating music similar to that of Frank Ocean. Furthermore, this process could be applied to any recording artist to generate music similar to theirs. In the future, researchers could maybe invest more resources into evaluating the models using subjective methods, such as a Turing Test.

REFERENCES

[1] Pitchfork, M. Hogan, M. Schnipper, S. Pearce, Q. Moreland, K. Lozano, J. Bromwich, J. Greene, R. Dombal, and L. Zoladz, "Frank ocean," 2017. [Online]. Available: https://pitchfork.com/artists/29508-frank-ocean/

[2] R. Kulkarni, R. Gaikwad, R. Sugandhi, P. Kulkarni, and S. Kone, "Survey on deep learning in music using gan," *International Journal of Engineering Research Technology (IJERT)*, vol. 8, no. 9, pp. 646–648, 2019.

[3] S. Gautam and S. Soni, "Music composition with artificial intelligence system based on markov chain and genetic algorithm," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 6, no. 2, pp. 1202–1205, 2018.

[4] M. Bretan, G. Weinberg, and L. Heck, "A unit selection methodology for music generation using deep neural networks," ser. Technical Report. The Georgia Institute of Technology, 2016.

[5] N. Kotecha and P. Young, "Generating music using an lstm network," ser. Technical Report. Columbia University, 2017.

[6] M. Marsden and R. Ajoodha, "Algorithmic music composition using probabilistic graphical models and artificial neural networks," ser. Technical Report. University of the Witwatersrand, 2021.

[7] N. Jaques, S. Gu, R. E. Turner, and D. Eck, "Generating music by fine-tuning recurrent neural networks with reinforcement learning," in *Deep Reinforcement Learning Workshop, NIPS*. IEEE, 2016.

[8] F. Colombo, A. Seeholzer, and W. Gerstner, "Deep artificial composer: A creative neural network model for automated melody generation," in *International Conference on Evolutionary and Biologically Inspired Music and Art*. Springer, 2017, pp. 81–96.

[9] D. Eck and J. Schmidhuber, "A first look at music composition using lstm recurrent neural networks." Instituto Dalle Molle di studi sull' intelligenza artificiale, 2002.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 12, p. 1735–1780, 1997.

[11] S. Sigurur, "How to generate music using a lstm neural network in keras," 2017. [Online]. Available: https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5

[12] D. Iourovitski, "Generating text with deep learning," *Medium*, Mar 2017. [Online]. Available: https://towardsdatascience.com/generating-text-with-deep-learning-8d3ffec3305b

[13] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," 2020.

[14] B. McMahon, "Ai jukebox: Creating music with neural networks," 2021. [Online]. Available: https://medium.com/@cipher813/ai-jukebox-creating-music-with-neural-networks-1d9585c4d649

[15] L.-C. Yang and A. Lerch, "On the evaluation of generative models in music," *Neural Computing and Applications*, vol. 32, 05 2020.

[16] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," 2018.

[17] A. Lulit, "Why frank ocean is the greatest artist of this generation," 2019. [Online]. Available: https://tideline.news/2622/ae/why-frank-ocean-is-the-greatest-artist-of-this-generation/