
Music Generation - G12

Aryan Verma, Aayush Sahay, Rajvir Singh, Ishaan Tandel

Abstract

In this project report we tend to generate music using deep learning techniques such as feed forward neural networks. Our report has been divided into various sections as the problem definition where we tend to analyze our problem statement and the scope we are dealing with in this paper. Before that we also discuss the various research papers we have gone through and the findings we discovered in them owing to the fact that few of them were really insightful and provided us with great resources thereby aiding in solving our problem statement. The research papers had various types of techniques applied and so we were very much interested in carrying out our decided methodologies. We then move onto our problem statement which is being expressed both mathematically and defined in words as the requirement of our project. We have also discussed our problem formulation in the next points as our intuition to solve the issue we feel is being left out in the research papers we have studied. The methodologies to be discussed which are being a part of our project have also been discussed at length. We then sum up our paper with the much needed experiments and observations and the conclusions we derive from our findings.

1 Introduction

Deep learning has recently become a pacing domain and is now used regularly for classification and prediction tasks, such as image recognition, voice recognition or translation. Computer music has continued developing for the general public, if we consider, for instance, the GarageBand music composition and production application for Apple platforms (computers, tablets and cellphones), as an offspring of the initial Cubase sequencer software, released by Steinberg in 1989. The research domain in deep learning-based music generation has turned hot, building on this work using artificial neural networks to generate music (e.g. experiments by Todd in 1989 and the CONCERT system developed by Mozer in 1994), while creating an active stream of new challenges made possible thanks to the progress of deep learning. Let us also note the growing interest by some private big parts of digital media in the computer-aided generation of artistic content, with the creation by Google in June 2016 of the Magenta research project and the creation by Spotify in September 2017 of the CTRL. This is likely to contribute to blurring the line between music creation and music consumption through the personalization of musical content. There is constant demand for new musical content for a multitude of users, ranging from artistic expression, to jingles for new TV shows, to elevator music.

2 Related Work

Deep learning is gaining traction as a method for music creation in addition to typical tasks such as prediction, classification, and translation, as seen by recent research groups such as Magenta at Google and CTRL (Creator Technology Research Lab) at Spotify. The objective is to use deep

learning architectures and training approaches to learn musical genres from arbitrary musical corpora automatically and then create samples from the predicted distribution. However, using deep learning to produce material directly quickly hits its limitations, since the generated content tends to mirror the training set rather than displaying actual innovation. Furthermore, deep learning systems do not provide direct control over generation (e.g., imposing some tonality or other arbitrary constraints). Furthermore, deep learning architectures by themselves are autistic automata that make music without human user involvement, far from the goal of supporting artists in the composition and refinement of music. Their investigation focused on issues such as control, structure, creativity, and interaction. In this paper [Music Generation by Deep Learning], they examined some of the drawbacks of a direct application of deep learning to music production, as well as why they haven't been addressed and how to overcome them using various techniques. As examples some of the of potential directions are discussed in this literature.

Although direct application of deep learning architectures and methods to generation may generate impressive results, it has significant limits. In this section, we consider:

- Control: tonality conformity, maximum number of repeated notes, rhythm, and so on.
- Structure: as opposed to rambling music with no sense of direction.
- Creativity: as opposed to imitation and the possibility of plagiarism.
- Interactivity: as opposed to automated single-step creation.

OpenAI's Jukebox (1), a music generating model which generates multi-track polyphonic music with rudimentary singing across different genres in raw audio files. It achieves this by using Vector Quantized Variational Autoencoder (VQ-VAE-2) at 3 different temporal scopes for compressing the audio domain and extracting the features from raw training audio. Then sampling is done from higher temporal scope to lower enhancing the quality of information sampled then these are decoded giving a generated audio output. MuseGAN (2) uses applying generative adversarial networks (GANs) to generate multi-track pop/rock piano-roll music of four bars, using convolutions in both the generators and the discriminators. This literature proposes 3 models for modelling multi-track interdependency being jamming mode, composer mode and hybrid mode. The work is able to generate from scratch and accompanying a track given as input.

This paper (3) focuses on methods for efficient large-scale sequence comparison and tests them on the specific problem of matching scores to corresponding audio recordings. In addition, by developing an effective system for matching a MIDI file to its corresponding entry in a database of audio recordings, this paper is able to leverage the large collection of MIDI files to create valuable ground truth information for content-based MIR. Throughout, the paper will focus on learning-based methods, algorithms which are optimized to achieve high performance over a collection of exemplary data.

This paper(4) 2-layered Long Short Term Memory (LSTM) recurrent neural network (RNN) architecture to produce a character level model to predict the next note in a sequence. Two experiments are run in this paper, a midi data experiment, where a midi message is treated as a single token, and a piano roll experiment, where each unique combination of notes across all time steps is treated as a separate token. The music is generated by feeding a short seed sequence into their trained model. They were able to make music that has both harmony and melody and is passable as music composed by humans and showed that a multi-layer LSTM, character-level language model applied to two separate data representations is capable of generating music that is at least comparable to sophisticated time series probability density techniques prevalent in the literature.

This paper (5) studies some Deep learning models and uses a Tied Parallel Network, which is a combination of a recurrent and a feedforward network. Their model is modified to generate songs in such a manner that the rhythm is controlled accordingly to the speed of a person. One application an MP3 streaming server and an Android app that tracks the speed by GPS is also made for implementing the model in real life application

86 Hadjeres and Nielsen suggest Anticipation-RNN (6) as a method for generating melodies with unary
87 limitations on notes (to enforce a given note at a given time position to have a given value). The
88 disadvantage of utilizing a typical note-to-note iterative technique for recurrent network development
89 is that, applying the restriction at a certain time step may invalidate the distribution of previously
90 produced items, as illustrated in (7). The goal is to condition the recurrent network (RNN) on some
91 information summarizing the set of additional (in time) constraints in order to predict impending
92 constraints and create notes with the right distribution.

93 Sun's (8) DeepHear architecture is designed to generate ragtime jazz tunes. The design consists of
94 four layers of stacked autoencoders (four hierarchically layered autoencoders), with a decreasing
95 number of hidden units, down to sixteen. The model is first trained on a corpus of 600 measures of
96 Scott Joplin's ragtime music, divided into 4-measure pieces. In order to generate an output (in the
97 same 4-measure long format as the training examples), random data is fed into the 16 bottleneck
98 hidden layer units as the seed and then feedforwarded into the chain of decoders.

99 3 Problem Description

100 We aim to generate piano instrumental melody with our model. For this we use symbolic
101 representation to represent each key by a symbol or label. We use a time step temporal scope with the
102 length being set to the smallest note duration possible to slice the training data and generated data
103 with respect to time. Let X_t represent the set of active notes at time step t . Let M represent a melody
104 then a melody can be represented in terms of sequence of X_t as,

$$105 \quad M = X_t \text{ for all } t \text{ belonging } t_0 \text{ to } t_n$$

106 Where t_n is the last time step of the melody.

107 Our problem statement can be defined as, given a set of melodies, train a model such that it can
108 generate a never ending melody. This can be divided into 2 phases, for the training phase the model is
109 given the set of melodies in the above representation and the model tries to learn the pattern between
110 the sequences of active notes, and in generation phase where the model tries to predict the next set of
111 active notes being given some previous sets of active notes.

$$112 \quad X_t = \text{generate}(X_{t-1}, X_{t-2} \dots X_{t-inseq})$$

113 where $inseq$ is the number of previous sets of note given, and $t - inseq < 0$

114

115 4 Problem formulation

116 By giving the training data of sequence of the set of active notes as training data we believe the
117 model can learn the patterns in sequence. The set of active notes in a time step can be represented by
118 using many hot encoding and feed into the model. It can then use this learned pattern to predict the
119 next node given a seed sequence of notes. By feeding back the predicted note in the sequence and
120 removing the oldest ones, it should be possible to create a non-ending melody iteratively. We can
121 approach this in two ways, either train a deterministic model making a one to one relation with the
122 given seed and the melody produced or a probabilistic model which selects one of the many possible
123 notes depending on probability it learned during the training phase.

124 5 Methodology

125 1. Libraries used:

- 126 • Pypianoroll (9) It is an open source library for working with piano rolls, it provides essen-
127 tial tools for handling multitrack pianorolls. It helps in manipulating, visualizing, saving
128 and loading multitrack piano rolls in a space efficient manner. It helps in parsing MIDI
129 files into multitrack piano rolls and write multitrack piano rolls into MIDI files.
- 130 • Keras It is also an open source library that provides python interface for artificial neural
131 networks. It supports multiple backend technologies including tensorflow, PlaidML.

It focuses on being user friendly and modular. It contains various implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers and a host of other tools to make working with image and text data easier to simplify the coding necessary for writing deep neural networks code.

2. Feed Forward Neural Network: A feed forward neural network is an artificial neural network wherein connections between the nodes do not form a cycle. It is different from the recurrent neural network. The feedforward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction—forward—from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network. More generally, any directed acyclic graph may be used for a feedforward network, with some nodes (with no parents) designated as inputs, and some nodes (with no children) designated as outputs. These can be viewed as multilayer networks where some edges skip layers, either counting layers backwards from the outputs or forwards from the inputs. Various activation functions can be used, and there can be relations between weights, as in convolutional neural networks. A single-layer neural network can compute a continuous output instead of a step function. A common choice is the so-called logistic function. Multi-layer networks use a variety of learning techniques, the most popular being back-propagation. Here, the output values are compared with the correct answer to compute the value of some predefined error-function. By various techniques, the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculations is small. In this case, one would say that the network has learned a certain target function. To adjust weights properly, one applies a general method for non-linear optimization that is called gradient descent. For this, the network calculates the derivative of the error function with respect to the network weights, and changes the weights such that the error decreases (thus going downhill on the surface of the error function). For this reason, back-propagation can only be applied on networks with differentiable activation functions.

3. Softmax Activation Function: Softmax is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector. The most common use of the softmax function in applied machine learning is in its use as an activation function in a neural network model. Specifically, the network is configured to output N values, one for each class in the classification task, and the softmax function is used to normalize the outputs, converting them from weighted sum values into probabilities that sum to one. All values are marked 0 (impossible) and a 1 (certain) is used to mark the position for the class label. For example, three class labels will be integer encoded as 0, 1, and 2. Then encoded to vectors as follows: Class 0: [1, 0, 0] Class 1: [0, 1, 0] Class 2: [0, 0, 1] This is called a one-hot encoding. It represents the expected multinomial probability distribution for each class used to correct the model under supervised learning. The softmax function will output a probability of class membership for each class label and attempt to best approximate the expected target for a given input. For example, if the integer encoded class 1 was expected for one example, the target vector would be: [0, 1, 0] The softmax output might look as follows, which puts the most weight on class 1 and less weight on the other classes. [0.09003057 0.66524096 0.24472847] The error between the expected and predicted multinomial probability distribution is often calculated using cross-entropy, and this error is then used to update the model. This is called the cross-entropy loss function. Although direct application of deep learning architectures and methods to generation may generate impressive results, it has significant limits. In this section, we consider:

- Control: tonality conformity, maximum number of repeated notes, rhythm, and so on.
- Structure: as opposed to rambling music with no sense of direction
- Creativity: as opposed to imitation and the possibility of plagiarism

- Interactivity: as opposed to automated single-step creation.

5.1 Required Steps

5.1.1 Dimensions of control strategies

Because ordinary neural networks are not meant to be controlled, such arbitrary control is a problematic challenge for existing deep learning architectures and methodologies. In contrast to Markov models, which feature an operational model where restrictions may be attached to their internal operational structure to govern the generation, neural networks do not have such an operational entry point. Furthermore, the scattered form of its representation does not allow for a clear correlation to the structure of the created material. As a consequence, the solutions we shall examine for managing deep learning production must depend on some external intervention at multiple entrance points (hooks), such as:

- Input
- Output
- Encapsulation/reformulation.

5.1.2 Sampling

If we set limits on the output creation, sampling a model to produce content might be an entrance point for control (this is called constraint sampling). This is often accomplished by a generate-and-test technique, in which valid solutions are chosen from a collection of created random samples from the model. As we shall see, a critical challenge is how to steer the sample process in order to meet the goals (constraints), hence sampling will often be supplemented with other tactics.

5.1.3 Conditioning

Conditioning technique (also known as conditional architecture) is to condition the design on some additional conditioning information, which might be arbitrary, such as a class label or input from other modalities.

5.1.4 Input Manipulation

DeepDream [MOT15] pioneered the input modification approach for pictures. The notion is that the original input material, or a completely new (randomly produced) input content, is changed progressively in order to meet a target attribute. It is important to note that generation control is indirect since it is applied on the input rather than the output before generation. Examples include:

- maximizing the activation of a specific unit, to exaggerate some visual element specific to this unit, in DeepDream (5)
- maximizing the similarity to a given target, to create a consonant melody, in DeepHear (8)
- maximizing both the content similarity to some initial image and the style similarity to a reference style image, to perform style transfer

Surprisingly, this is accomplished by utilizing normal training processes, notably back-propagation to calculate gradients and gradient descent to reduce cost.

5.1.5 Reinforcement

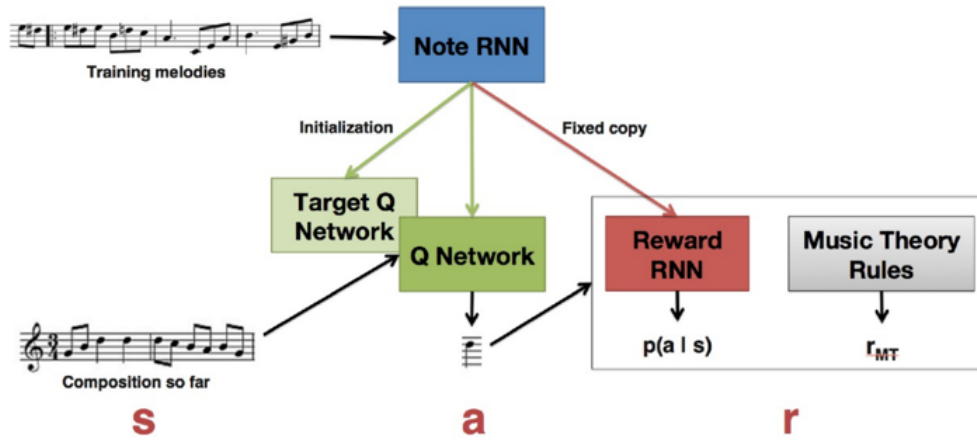
The reinforcement strategy ?? involves reformulating the generation of musical content as a reinforcement learning problem, with the output of a trained recurrent network serving as an objective and user-defined constraints, such as some tonality rules based on music theory, serving as an additional objective. Following are the steps of included in reinforcement learning:

- An agent picks and executes actions sequentially inside an environment.

- Each action completed leads it to a new state
- with feedback (from the environment) of a reward (reinforcement signal), which reflects some adequacy of the action to the environment (the circumstance).

The goal of reinforcement learning is for the agent to learn a near optimum policy (sequence of behaviors) to maximize its cumulative rewards (named its gain). The generation of a melody may be expressed in the following way (as shown in following Figure): the state s represents the musical content (a partial melody) created so far, and the action a indicates the selection of the next note to be generated.

Figure 1: Reinforcement



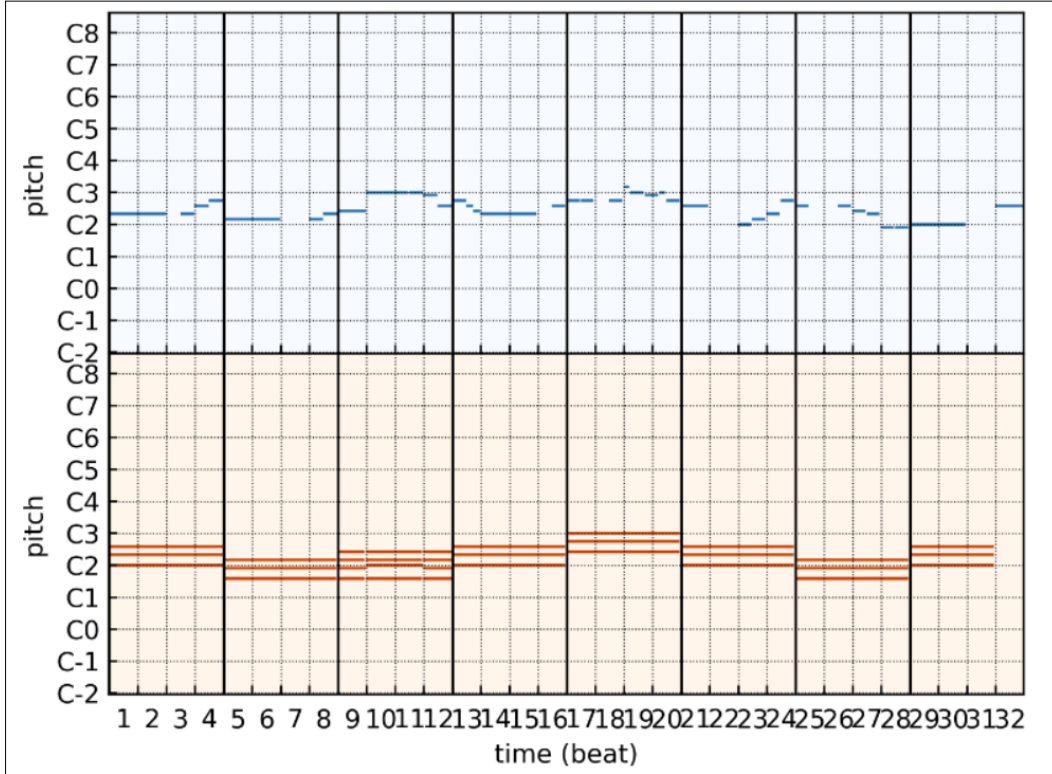
5.1.6 Unit Selection

The unit selection approach is based on querying and concatenating consecutive musical units (e.g., a melody inside a measure) from a database in order to construct some sequence based on certain user criteria.

6 Experiment

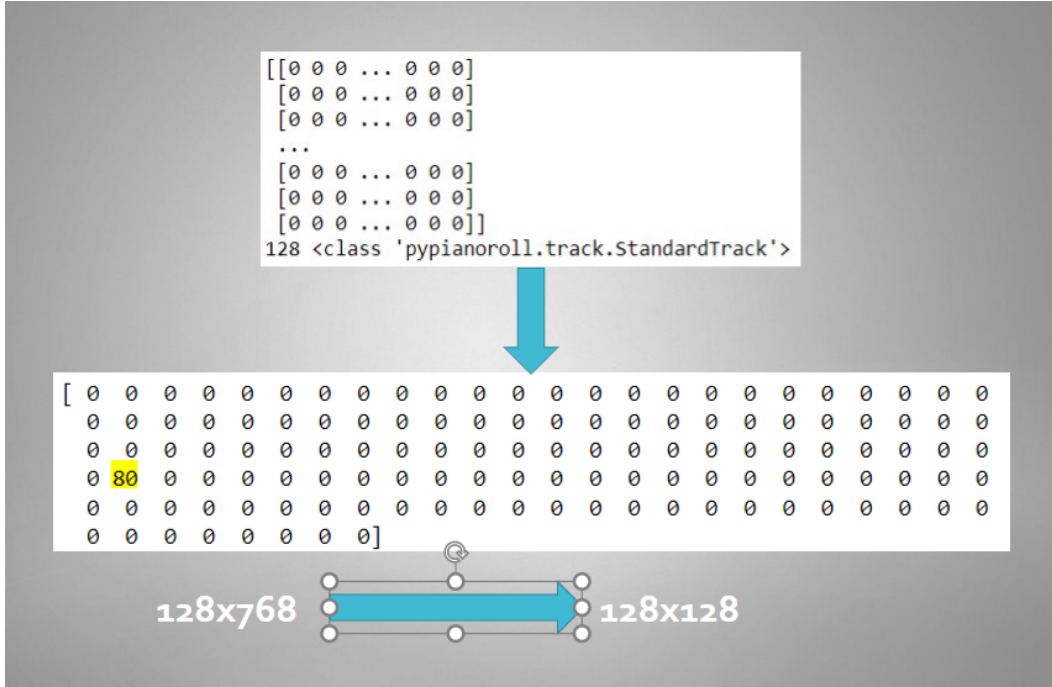
We used the Lead Sheet Dataset (10) rather than the Lakh pianoroll dataset 2 as we wanted to start with monophonic music and this dataset gives us most of the widely known music archives rather than the repetitive medley music as given in Lakhpianoroll. This contains around 11 thousand songs from around 5 thousand different artists. It has a total of 18 thousand leadsheets including the chorus and verse of different song. The beat resolution or the temporal resolution of this data set is set to 4 with a 4/4 time step. The whole music sheet is divided in 128 notes. Our approach towards solving this problem is to do the preprocessing of the dataset we have to match our needs. Then training our keras sequential ANN model with the given data. Then we generate a newer note or the 9th note using the feed forward network. The last step involves preprocessing to convert the array like modifiable structure back to playable MIDI format. In preprocessing, first we convert the MIDI files to pianoroll representation. Then we compress the dataset and change it to a more usable design according to our specific needs. Then we one hot encoded the data based on the 128 note representation so that it can be used to predict the note in the next time step.

Figure 2: Dataset



Our compression scheme compresses the 128*768 data to 128*128 data. We are sampling every 6th note after analyzing most of the songs and finding out that most songs use 6 note beats. The diagram in 3 shows 1 whole track and the diagram below shows how 1 time step looks under a track. We can see 1 note active the given time step . There were 768 of these in one track and we reduced that to 128 arrays in 1 track . Hence decreasing the resolution of our track's from 24 to 4 but at the same time using 6 times lesser space in total . Also it results decreasing the training time. We transposed the given dataset randomly over a range of 66 to 84 in both upward and downward direction so that our model is trained for each type or scale of music and can work accordingly. As most of the sample data were in the range of c3 and c4. We tried to transpose them till c8 and c1 so that our model can handle any type of use case. This transpose is random for each sample and is used to change the scale of the given song. As here in the 4 we can see that the song on C4 pitch I shifted as it is to the c5 pitch. The model is first trained on these 18 thousand midi files with random scale on 100 epoch. Then, a 2 layer feed forward network is used with 8x128 input and 128 outputs. The softmax activation function then assigns a probability to each note after observing the previous 8 notes. Then the most probable note is chosen as the next node from the trained model. This is then appended to the current track and this procedure is repeated for 8 notes. then, the 9th note is a random not from the list of 3 most probable note. Hence inducing some randomness in the music. This sometimes also results in noise rather than a variation in music. The loss function is a categorical cross entropy function with total 131,200 parameters. Now, in the post processing, we are transposing the track mean pitch back to C3 so that it doesn't sound too high or low pitched. At last, the pianoroll is converted back to the playable MIDI format using the pypianoroll library.

Figure 3: Compression Scheme



7 Result and Conclusion

We've developed a model that generates raw audio music that imitates a wide range of styles and performers. We can condition this music on various artists and genres, as well as specify the sample's words. We laid out all of the details needed to effectively train and compress the music into tokens. While earlier research has produced raw audio music in the 20–30 second range, our algorithm can produce compositions that are several minutes long and feature recognisable singing in natural-sounding voices. We were able to demonstrate a character-level language model applied to two independent data representations may produce music that is at least similar to advanced time series probability density techniques commonly used in the literature. Our models were shown to be capable of learning significant musical structure. The first plot is being the pitch v time graph of the sound frequencies developed without choosing it probabilistically and the second one was with plotted choosing probabilistically and this was done to show the difference occurred to avoid unnecessary looping of pitch that occurred inferring from every 8th note and thus avoiding the same. The results being operated here are giving a resonating frequency sounds that may or may not compulsorily end up giving melodious symphony rather just a sequence of similar pitched voices. This needs to be improved and the same is to be work under in future. Currently the results we got in music generation are that satisfactory and certain amendments are necessary.

8 Future Work

We will work to demonstrate that a multi-layer LSTM character-level language model applied to two independent data representations may produce music that is at least similar to advanced time series probability density techniques commonly used in the literature and making our models capable of learning significant musical structure. Increase the number of input notes to 8 which will be used to create next 4 notes. The main reason for doing the mentioned work is to reduce redundancy and enhance the quality of voice generated with excelled accuracy. This will also create possibilities to improve synchronization in the generated music and will be more efficient in producing good quality

Figure 4: Random Scale Transpose

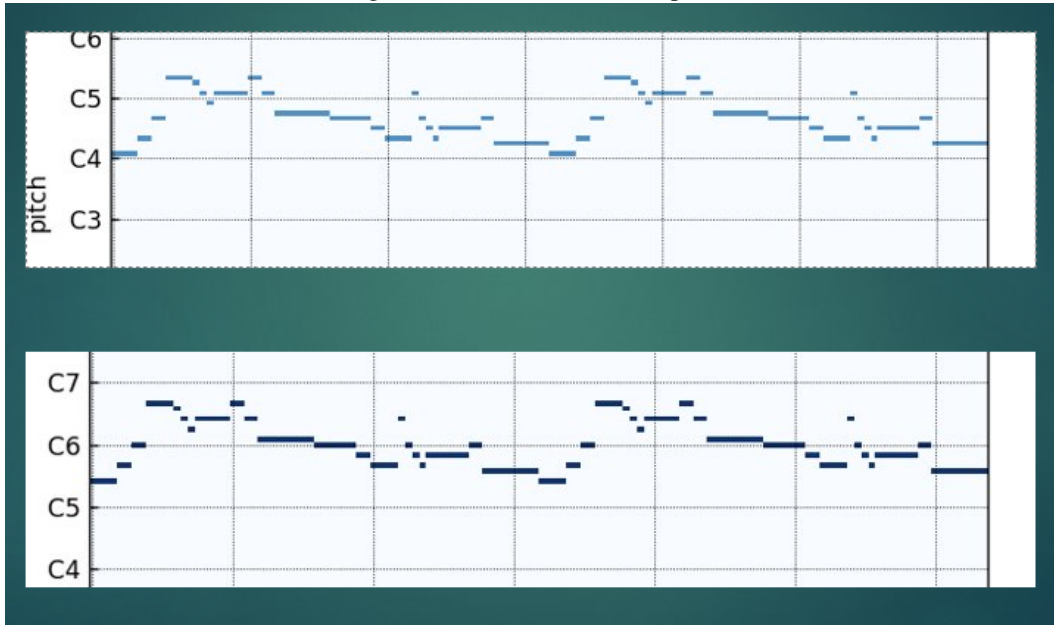
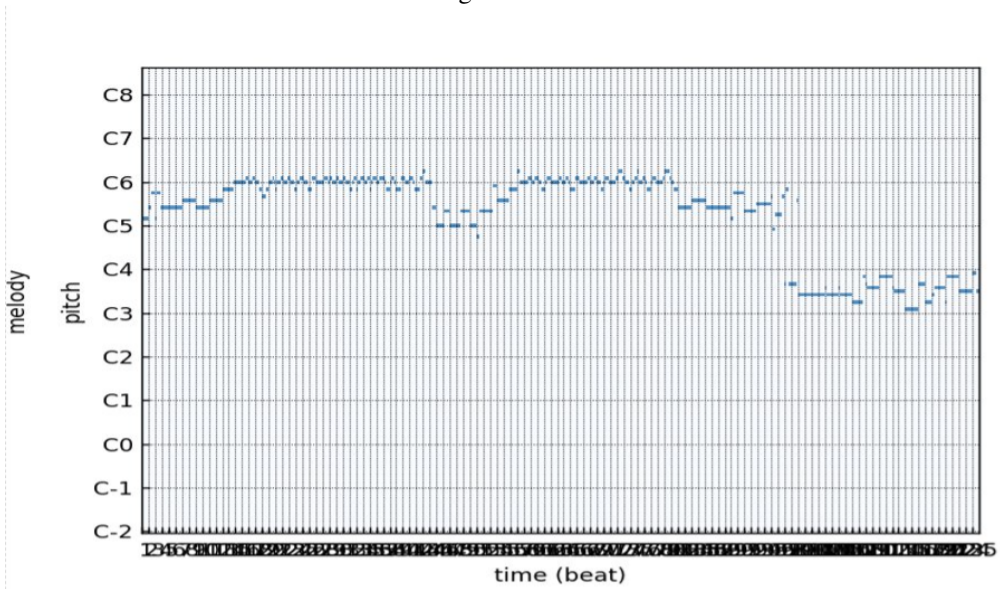
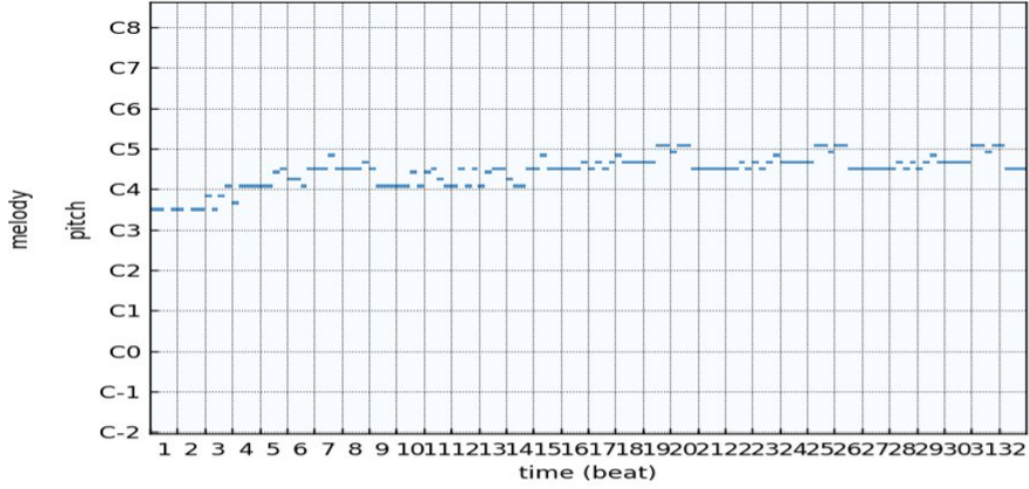


Figure 5: Result 1



305 music. We intend to continue our work by adopting more complex models and data representations
 306 that successfully capture the underlying melodic structure, given the recent passion for machine
 307 learning inspired art. Furthermore, we believe that more work could be done to design a stronger
 308 method for assessing a piece's quality — only then will we be able to train models capable of properly
 309 composing original music.

Figure 6: Result 2



References

- [1] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, “Jukebox: A generative model for music,” *arXiv preprint arXiv:2005.00341*, 2020.
- [2] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [3] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.
- [4] A. Huang and R. Wu, “Deep learning for music,” *arXiv preprint arXiv:1606.04930*, 2016.
- [5] V. A. A. de Souza and S. E. F. de Avila, “Deep neural networks for generating music,” 2018.
- [6] G. Hadjeres and F. Nielsen, “Interactive music generation with positional constraints using anticipation-rnns,” *arXiv preprint arXiv:1709.06404*, 2017.
- [7] F. Pachet, P. Roy, and G. Barbieri, “Finite-length markov processes with constraints,” in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.
- [8] F. Sun, “Deephear-composing and harmonizing music with neural networks,” URL: <https://fephsun.github.io/2015/09/01/neural-music.html>, 2017.
- [9] H.-W. Dong, W.-Y. Hsiao, and Y.-H. Yang, “Pypianoroll: Open source python package for handling multitrack pianoroll,” *Proc. ISMIR. Late-breaking paper*; [Online] <https://github.com/salu133445/pypianoroll>, 2018.
- [10] S. F. T. K. B. G. H.-M. L. H.-W. D. Y. C. T. L. Yin-Cheng Yeh, Wen-Yi Hsiao and Y.-H. Yang, “Automatic melody harmonization with triad chords: A comparative study,” *Journal of New Music Research (JNMR)*, vol. 50, no. 1, pp. 37–51, 2021.