Using a Language Model to Generate Music

Austin Gongora
DS340W Section 001
College of IST
State College, PA
aag5363@psu.edu

Lily Magliente
DS340W Section 001
College of IST
State College, PA,
llm5423@psu.edu

***Abstract*—Music Composition has long been a joy of human pass time. With the advent of pattern recognition tools such as machine learning, humans have the ability to further explore their relationship with music. Music, for all its noise and variety of instruments has an actual structure associated to it. Music is composed of tones, rhythm, notes, and instruments that belong to different families. MIDI files represent a compressed musical composition sheet that can be interpreted by a computer. We aim to use MIDI files to learn the structure of music which facilitates the process in generating new sounds. We propose a methodology to leverage machine learning to further explore the capability of human creativity in regards to music creation. Here we show a process of generating music by using a language model. Although MIDI files do a great job at communicating to a computer the keys and notes of a musical sheet, there are still many more tools and software required to generate music that is comparable to human generated music. These tools include a MIDI synthesizer and software such as Sonic Visualizer for editing and creating MIDI files, and ultimately a more specific domain knowledge about audio engineering. Nonetheless, given our expertise and tools, we managed to generate noise that is perhaps the first time such a noise has ever been heard. As exciting and innovative as Artificial Intelligence (AI) generated music might seem, the scope of the problem extends beyond solely the data and the algorithm, but also other tools and software that can be at times very unfamiliar with the researcher. Generating music sets forth an interesting challenge in that this project must generate musical composition and then add a statistical approach to determine the outcome of sequences for the composition.**

***Index Terms*—music generation, GPT-2, audio engineering, MIDI**

I. Introduction

The cross sections of creativity and machine intelligence is an interesting focal point for research and Artificial Intelligence (AI) communities. A quick glance at Tensor Flow's Magenta [3] highlights the amount of time and effort spent in developing creative AI tools. This project allows the opportunity to explore how AI can be leveraged for creativity and passion. Personalization is becoming an important component for any internet company; whether it be in the form of a curated daily Spotify music playlist, customized news feeds on Twitter, or recommended videos on YouTube, personalization improves a user's experience. The question at hand is: How can one go beyond a curated list of items and create something truly personalized for a user? Can one generate user-specific content on internet entertainment platforms? The team's motivation is to create a framework that has the ability to generate music that is convincing enough to pass as a human creation.

On top of this, are we able to accomplish this task by using a language model that was originally trained for dealing with natural language processing tasks? Can our team create a robust approach that handles audio files as if they were language tasks? Natural language processing tools have exploded with new techniques and methods. These tools include BERT, GPT-2 and Google Translate, to name a few examples. How powerful are these tools and can they be used for tasks outside of their original scope? We are interested in exploring these questions and proposing a new approach to how we treat language models in a machine learning pipeline.

II. Approach

The focal point of our approach is a generative language model. Traditionally, language models have been primarily used for language tasks such as speech recognition, handwriting recognition, Optical Character Recognition and other natural language processing tasks. Language models purpose is to learn the rules of a given language and provide context to the researcher. We can think of these rules as a system of symbols that coherently work together to form a grammatical structure that is shared and agreed among those that use the language. Typically, we define a language by its dialect rather its context. The context however, is important to consider when developing language models that solely rely on the information that is provided.

Allow us to form an example to focus on the point at hand. Let us consider two distinct corpuss: one of Hollywood critic reviews and another of medical surgery notes. Now let us consider the word, "extract". Without given any context of the language used it would be difficult to discern the subject matter. We focus on this example as it mirrors our own work with language models. Our work is highly dependent on the relationship that language has in regards to context and language generation.

With respect to the language generation task, we used a state-of-the-art unsupervised model - GPT-2 - developed by OpenAI in February of 2019 [4]. In short, GPT-2 was trained on 40 GB of internet text data with the primary focus of predicting the next word in a sentence. GPT-2 has outperformed other gold-standard language models on popular language benchmarks such as LAMBADA [17] and Winograd Schema [18]. What makes this technology unique lies in the algorithm's ability to generate conditioned sequential information that retains the topic and structure of the text. Sequence-to-Sequence models leverage statistics by assigning a probability likelihood for a given word or in some cases a collection of words. By utilizing prior information, sequencing models use likelihood probabilities to generate a proceeding sequence. A major criticism of using this category of language models is in their ability to quickly lose the original context of the token. We take this complication into account and discuss different approaches we apply to counter some of these effects.

The capability for GPT-2 to be a language model that can capture the context of a corpus with such limited data makes the algorithm a perfect candidate for our project- that is we do not need to do task or language specific training to garner preliminary results. This unique characteristic allowed us to perform experiments in an unfamiliar approach. We are interested to test how the algorithm works with non-traditional language. Specifically, our group focuses on *Musical Instrument Digital Interface*, or **MIDI**, type files. More on the composition and logic of the MIDI structure under **Data**. Reference figure 4 to see how a computer interprets a MIDI file. The important takeaway is that a MIDI file is a combination of numbers and letters with an inherent structure associated with it. Are able to leverage the strengths of GPT-2 to generate sounds using existing MIDI files? Success for our team comes in the form of either a continuation of the existing MIDI file through GPT-2 or by creating a new MIDI file all together.

III. Related Work

For the past few decades, researchers and artists have found themselves at an intersection of machine intelligence and the fine arts. In particular, we are interested in delving deeper into understanding the relationship between machine learning and musical compositions. One of the earlier instances of researching such a relationship is shown in David Cope's research in 1981, Experiments in Musical Intelligence (EMT) (Moura, 2018). Cope's research in EMT was rooted in recobinancy, meaning he aimed to create the new music by "recombining extant music into new logical successions" [10].

Flash forward three decades, the private company Mubert, focusing on Artificial Intelligence (AI) music solutions, has extended their research to the entertainment industry. The company has shown a clear use case that appeals to over thousands of users. Mubert offers multiple genres such as house, techno, and ambient as well as a variety of activities ranging from work to relax mode.

The term generative music was coined by musician Brian Eno in 1995. Eno described an autonomous system that would never sound the same way twice. The idea was to create a more personal listening experience for the audience. The idea was for the system to follow on overarching musical and incorporate randomness. Two of Enos most prominent works include Scape [14] and Bloom [15] which are mainly used by the video gaming industry.

More recently in 2019, a research project, GanSynth aimed to create music using GANS. The model runs quickly to create parallel sequences, which ultimately generates the music. The efforts are focused on using Progressive GANs, which are an extension from GANs, as it progressively grows both the discriminator and generator, by adding layers and details. This improved the frequency resolution of the short-time fourier transforms boosts performance as it separated the harmonics.

Our project was derived through the variety of processes and ideas previously mentioned. While GanSynth and Mubert generated music with electronic and modern beats, we wanted to discover how a similar model would fair with regard to just single instruments at a time.

IV. Data

In this section, we will outline the process of obtaining the different data sources, in the correct format, which was required to develop the project. Throughout this semester, we have experimented with multiple approaches of how to obtain the data. We learned that working with audio files was more of a data engineering problem than we originally planned. The angles that we looked at to solve the issue of working with audio files was tenfold, as we attempted to convert audio files to midi, web-scraped and finally downloaded a MIDI repository.

A. WAV2MIDI

A .wav file, short for wave form, is an audio format that computers interpret as a sound file. The .wav file type was not suitable for development, so our next course of action was to attempt to convert the .wav file to a type of file that could be used for easy computation. After doing further reading on audio engineering that revolves around editing, creating, publishing audio files it was decisive to work on files that were in the MIDI format. More information on the MIDI format is found in the MIDI section, but essentially a MIDI file can be thought of as a compressed musical note sheet that is written in binary, allowing a computer to interpret the patterns and tones. Our ambition was to take existing .wav files and convert them into this format that could be interpreted. The proposition seemed promising, allowing us to leverage any .wav file we wanted to use as our test cases.

The process to create new MIDI files required downloading the musical software Sonic Visualizer [16]. Sonic Visualizer is an open-source tool intended for audio researchers to study music recordings in an easy to use manner. Aside from Sonic Visualizer, it was also required to use a Vamp plugin to return detailed information on audio data. Plugins are used for audio feature extraction and in order to properly convert a .wav file to MIDI these are some of the tools that would be required. In easier terms, together these tools extract the melody of a given song and the frequency that relates to the pitch of a song.

Python packages such as jams, MIDIUtil, vamp, and SoundFile are available to streamline the process of conversion. This process of obtaining .wav files is suitable when wanting to choose a specific MIDI file type, but there are its own downfalls associated with this process. For example, most audio engineering pipelines rely on hardware such as MIDI synthesizers or MIDI keyboards to properly construct the MIDI file. Without these tools, any output is susceptible to corruption or misinterpretation of the .wav file.

B. Groove MIDI Data

The Groove MIDI Data set was taken from the Magenta website. It includes an easily downloadable zip file of 13.6 hours of music in MIDI format. This has about 22,000 measures of music. A measure is also known as a bar and it has a specific number of beats, corresponding to the beats per minute (BPM) of the song. The audio files are by professional drummers who were instructed to improvise long segments of songs. There is a well organized .csv of information for each individual music sample in the data. Each row is a recording session and the corresponding id numbers for the session, and drummer. There is an attribute called style, which is the genre of the music. The BPM, time signature, duration and split type information is also all included in the .csv.

This MIDI data set is the basis of this project. We plan to utilize the MIDI data to generate new MIDI files (see Section Using a Language Model to Generate Music to learn more about our model). The whole data set is comprised of drum music. As mentioned before, we initially wanted to focus on Deep House and Jazz music, but now we will focus on drumming music because of the convenience of the data set. We planned to backtrack after we built a model that outputs representative music given the input to specialize in other music genres.

We uploaded all of this data to XSEDE. Given the format, we needed to upload it using Globus, because we got errors using just the terminal. We downloaded Globus Connect Personal and created a collection that read into the local computer. Then we transferred all of the data folders to XSEDE via Globus.

C. Webscraping

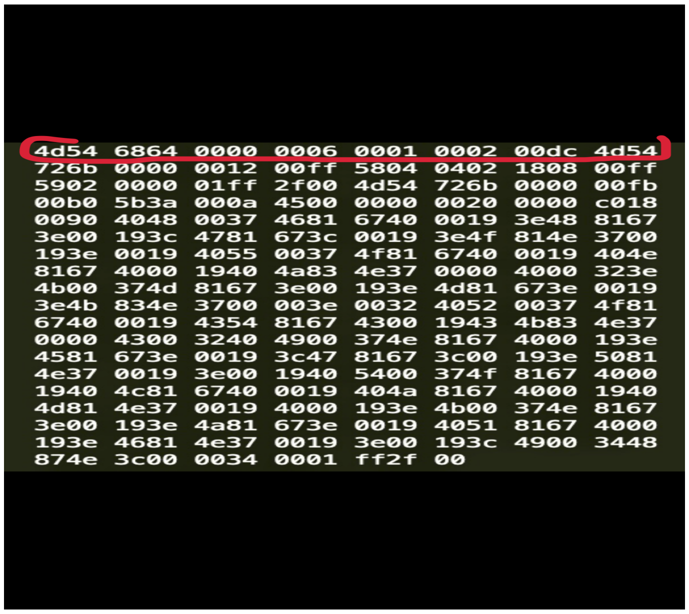
We obtained the data from multiple sources by either directly downloading midi files or using web-scraping techniques. We used the web-scraping Python package, Beautiful Soup to download the data.

Through open source MIDI developer, Bernd Kreuger, we scraped his midi repositories for piano specific files corresponding to prominent composers. We used a similar approach to scrape additional MIDI files from the site MFiles, which offered a diverse range of music in the midi format, like classical and traditional. This gave us hours of music to use as training data. Collectively, these sources offered us a sufficiently large and robust variety of music to create our model. Throughout the process of obtaining MIDI files, we made sure to remain consistent so we can have a better understanding of the target, which will ultimately generate new types of piano and drum sounds.

D. MIDI File Format

Musical Instrument Digital Interface, MIDI files, are compressed music sheets that provide instructions for the computer to interpret and output the corresponding noises. Each MIDI file has one or multiple MIDI streams, which provides musical characteristics about the tempo, beat, signature and more. The file is split up into two chucks: header and track, containing information about the file and the sequence stream, reference Figure 1 to see the MIDI header. The header has three 16-bit fields. The header specifies the organization, number of tracks and ticks per quarter note. The track chunk is where the actual song is stored and has a sequential stream of data containing MIDI information.

There are three different formats of MIDI, 0, 1 and 2. A different form does not change the way the MIDI file sounds but instead is differentiated by the number of tracks that follow the header. For MIDI File Format 0, there is one track chunk that follows. MIDI File Format 1 and 2 contain more than one simultaneous tracks or independent single track patterns, respectively. The data can be defined by the Second Byte Value, through binary, hexadecimal or decimal. MIDI files also need Garage Band, a Python script, or a third party software in order to be played. Garage Band is able to read and interpret the details, so when we used it to tun the MIDI file, it showed us the different instruments and notes that the song is comprised of. The MIDI file format is ideal for this project because of the level of complexity and details that it includes.



```
4d54 6864 0000 0006 0001 0002 00dc 4d54
726b 0000 0012 00ff 5804 0402 1808 00ff
5902 0000 01ff 2f00 4d54 726b 0000 00fb
00b0 5b3a 000a 4500 0000 0020 0000 c018
0090 4048 0037 4681 6740 0019 3e48 8167
3e00 193c 4781 673c 0019 3e4f 814e 3700
193e 0019 4055 0037 4f81 6740 0019 404e
8167 4000 1940 4a83 4e37 0000 4000 323e
4b00 374d 8167 3e00 193e 4d81 673e 0019
3e4b 834e 3700 003e 0032 4052 0037 4f81
6740 0019 4354 8167 4300 1943 4b83 4e37
0000 4300 3240 4900 374e 8167 4000 193e
4581 673e 0019 3c47 8167 3c00 193e 5081
4e37 0019 3e00 1940 5400 374f 8167 4000
1940 4c81 6740 0019 404a 8167 4000 1940
4d81 4e37 0019 4000 193e 4b00 374e 8167
3e00 193e 4a81 673e 0019 4051 8167 4000
193e 4681 4e37 0019 3e00 193c 4900 3448
874e 3c00 0034 0001 ff2f 00
```

Fig. 1. MIDI Header Format

E. Data Preparation

Creating an algorithm to generate music requires the research group to develop a deeper understanding of the fundamentals of music. Unfortunately, the approach is not as simple as feeding a neural network a lot of Beethoven and magically outputting a new song. Music which is both creative and expressive, contains a structure that consists of patterns and sequences. As a data science research group, we plan to

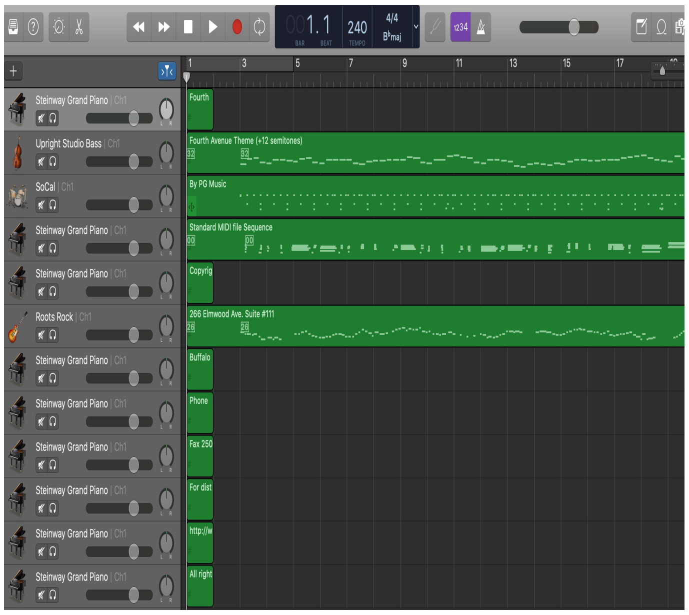


Fig. 2. Garage Band

leverage these patterns and sequences to formulate a workflow that can generate music. We have done extensive research into the formulation of our data pipeline, and after trial and error we have settled upon a logical workflow.

Moving forward with the MIDI file was the first major step of progression for our group. We had now acquired the type of data that was representative of the audio we intended to use to generate new sounds. Understanding the MIDI file and its contents posed a new challenge. Between the group there was minimal experience in audio engineering, so collectively it was necessary to further our knowledge on the details of the MIDI format. One observation we had was that all MIDI files shared the same MIDI header format, reference Figure 1. Mentioned in the MIDI section of the paper, the header format was a way of communicating to the computer that the following contents were of the MIDI type and should be treated as such. This was an assumption that was needed to be built into the logic of pipeline to properly create a MIDI file. Prior to applying this type of logic, we were unable to properly create any kind of output that was able to be interpreted as a MIDI file structure. Once we understood this crucial step we could begin to incorporate this to form new MIDI files.

Using GPT-2 to predict the next sequence of a MIDI file posed its own unique problems as well. GPT-2 is originally built as a language model, meaning that its primary focus is to be used in the context of the English language. Although GPT-2 is familiar with numeric structures, we often times found ourselves having to deal with actual text that was being predicted after a numeric sequence. A logic that had the ability to filter through outputs that were not in the MIDI format were necessary. The process was not entirely difficult. After understanding the MIDI format, we could filter out any output that contained phrases or words. Aside from this, we used logic that would check the format of the output of MIDI files. Reference Figure 3 and notice that the MIDI is a collection of 4 integers and characters with the exception of the last string that represents the ending of a MIDI file. We used this to our advantage and would only consider output that held this structure.

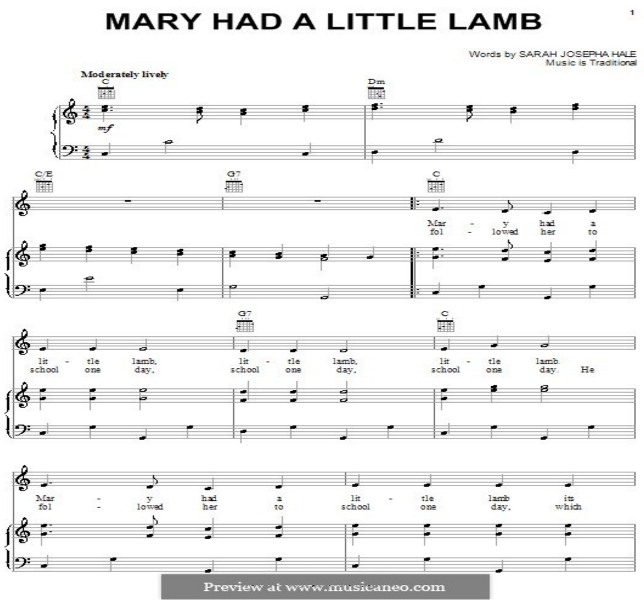


Fig. 3. Musical Composition

F. Musical Sheets

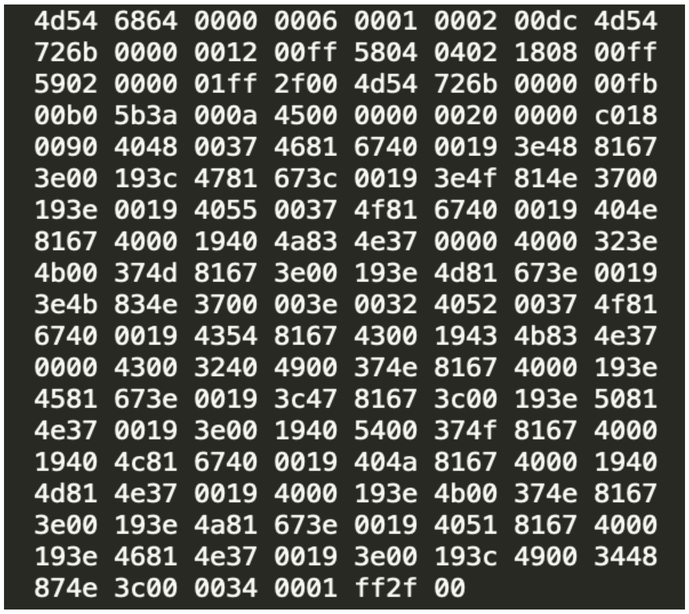
Figure 3 is an example of a musical sheet. Musical sheets are used by musicians as a way of communicating their work to produce music. They hold the structure and notes of music and allow for replication as long as the interpreter can read and play music for the given instrument. The sheet music specifies the tempo/speed of the song. Different notes are represented by their location on the lines and the length that each note is played for is also represented differently. This type of language can only be useful for those that have a deep understanding of the domain.

V. Using a Language Model to Generate Music

Our team propose a new method of music generation using an advanced language model. As described in the introduction we utilize GPT-2, an algorithm that aims to predict the next word in a sequence. In our specific case we are curious to test if a language model as advanced as GPT-2, is robust enough to treat MIDI files similarly to text files. Our goal is to generate either new sounds or a similar continuation of what the original audio files sounded like.

A. GPT-2

GPT-2 is a state-of-the-art language model released by OpenAI. This new technology was so powerful that the community was originally hesitant to release their fully trained 1.5 billion parameter model, stating they were unsure of the negative



```
4d54 6864 0000 0006 0001 0002 00dc 4d54
726b 0000 0012 00ff 5804 0402 1808 00ff
5902 0000 01ff 2f00 4d54 726b 0000 00fb
00b0 5b3a 000a 4500 0000 0020 0000 c018
0090 4048 0037 4681 6740 0019 3e48 8167
3e00 193c 4781 673c 0019 3e4f 814e 3700
193e 0019 4055 0037 4f81 6740 0019 404e
8167 4000 1940 4a83 4e37 0000 4000 323e
4b00 374d 8167 3e00 193e 4d81 673e 0019
3e4b 834e 3700 003e 0032 4052 0037 4f81
6740 0019 4354 8167 4300 1943 4b83 4e37
0000 4300 3240 4900 374e 8167 4000 193e
4581 673e 0019 3c47 8167 3c00 193e 5081
4e37 0019 3e00 1940 5400 374f 8167 4000
1940 4c81 6740 0019 404a 8167 4000 1940
4d81 4e37 0019 4000 193e 4b00 374e 8167
3e00 193e 4a81 673e 0019 4051 8167 4000
193e 4681 4e37 0019 3e00 193c 4900 3448
874e 3c00 0034 0001 ff2f 00
```

Fig. 4. MIDI Contents

consequences that could come with the release. GPT-2 has been a tool used on many different types of language tasks such as question answering, machine translation, and text summarization. GPT-2 represents a larger shift in the machine learning community that points to the rapid development of powerful models that are being incorporated for everyday use.

B. *Embeddings*

The power behind GPT-2 lies in the ability to generate new embeddings that are conditioned on existing embeddings. Embeddings represent data in a high dimensional space that accounts for the multitude of features and feature interactions that might not be so evident to researches. Embeddings work well for many different data types such as images, text, and location data. In this case we are representing our MIDI files as a 512-dimensional vector.

C. *Visualizing Embedding Space*

512 dimensions is incomprehensible to perceive and draw and extrapolations from the data. After even four dimensions, the relationships and features in any data become hard to grasp. To visualize and make sense of our data in a more appropriate manner we incorporate a method know as T-Distributed Stochastic Neighbor Embedding, or T-SNE for short. The idea is to take a collection of high dimensional data and apply dimension reductions techniques to garner a better understanding of what the data is communicating. We took the embeddings of 150 MIDI files, corresponding to 75 drum rolls and 75 piano compositions. We then ran this through our T-SNE to render a visualization. Reference Figure 5. T-SNE is a tool that can be used as an unsupervised approach to detect where clusters and relationships exist in the data. We can use the findings to extrapolate knew truths about the data or confirm existing truths. In this case we can confirm there is a difference between the MIDI structure of two instruments, piano and drums.

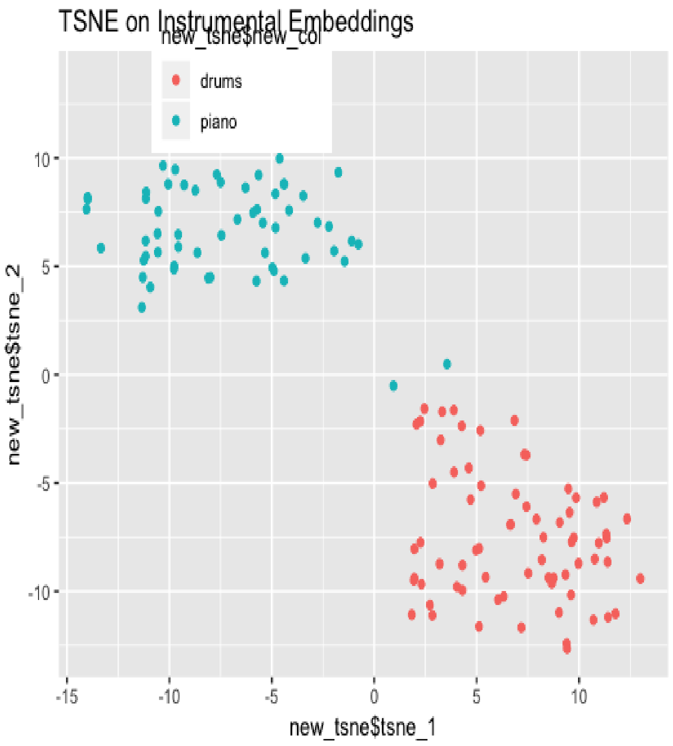


Fig. 5. T-SNE Drums Vs. Piano

VI. Evaluation Metrics

The question, how to evaluate this performance, comes to mind. This was an area that was originally difficult to comprehend because we were dealing with abstract data, in this case noise. How can we take a quantitative approach to how something sounds? We regarded sound and music as subjective. A popular quote summarizes this wonderfully, Beauty is in the eye of the beholder. We took this literally and found ourselves taking numerous approaches to determine the quality of our work. The earliest approach of evaluating our work was relying on the crowd to serve as judge. Approaching family and friends to critique different sounds that were either produced by us or harvested on the internet from sites like soundsnap. However, considering these sounds were short sound bites and were never appealing to the ear, we ultimately found this approach to lack real substance.

One approach that we are currently implementing is to compare the output of GPT-2 with the original MIDI file. This was not so clear to us in the beginning because as a group we were mostly focused on the low-hanging-fruit of generating sound regardless of what it may sound like. As we began to finish our project we noticed potential to develop a quantitative measure for evaluation. We could use the generated MIDI files and compare them with the original MIDI file data. We do not have a substantial amount of work done in this area but we have implemented it on some of the outputs that we have.

As of now we conclude that GPT-2 is not suitable enough to predict a sequence that is anywhere remotely close to the original output.

VII. Issues with Methodology

Issues with our project's methodology include the needing to develop a strong understanding of MIDI files and how to use them in the most efficient way from the beginning. We failed to know that we needed expensive and highly advanced technology until later in the project. For example, we needed a MIDI synthesizer and Keyboard to take the next steps to further the project. Also, given that GPT-2 is a language model, the sequences quickly lost effect. It produced usable output initially but then the predictions began to not make sense because language models work for sentences and words, not music. Working with audio files was much more of a data engineering issue than we initially thought. Working with MIDI files and attempting to leverage them to maximize our project took up a lot of the semester because we needed a good basis to produce a project.

VIII. Future Work

There are many avenues in which this project can be extend towards. Some ideas include evaluating the sentiment of the music which can be queued for listeners with respect to their activity. This could potentially be connected to a user's smart watch and if their watch senses that they are working out, then it would queue fast paced music that would accompany a workout. As seen by the following visualization, there is a natural split between music genres. We would use classification to classify the input data to output a similar sentiment of music. We would like to generate coherent sequences that rely on tone and rhythm so there could be music the people would actually like to listen to. The five second bits produced through this project would be good for a ringtone, but continuing to advance and elongate this would significantly improve the model. Adding more advanced metrics to the evaluation step would also further and separate our work from other. For example, if we used Amazon MTurks so people could manually label the produced noises and good or good. We can also train a model that could potentially produce lyrics. While we understand the difficulty of such a task because the lyrics would need to make sense and correspond to the patterns/format of a song, this next step in research would really set this project far apart from anything previously produced. Focusing on such a challenging task would have been rewarding because it would have shifted our focus further from what has been produced already, like music recognition and word predictors to merge them for a new project.

IX. Conclusion

Applying machine learning to a subject as creative and diverse as music opens the door for creativity and new innovations. Specifically, leveraging abstract data such as MIDI files is a promising avenue of success for future machine learning researchers and musicians alike. Through our work we have shown there to be a meaningful crossover between these two distinct areas of expertise. We hope that the interest and collaboration in this field will expand over time. We hope the concept of using AI to generate music becomes a common expectation for future music listeners.

X. Acknowledgements

We would like to thank Dr. James Wang for his assistance in helping us complete this project. Additionally, we appreciate our classmates who have also given helpful feedback to us as we developed this project.

References

[1] Goodfellow, Ian, et al. ”Generative adversarial nets.” Advances in neural information processing systems. 2014.
[2] Radford, Alec, et al. ”Language models are unsupervised multitask learners.” OpenAI Blog 1.8 (2019).
[3] https://magenta.tensorflow.org/
[4] https://openai.com/about/
[5] https://openai.com/blog/musenet/
[6] http://artsites.ucsc.edu/faculty/cope/experiments.htm
[7] Karras, Tero, et al. ”Progressive growing of gans for improved quality, stability, and variation.” arXiv preprint arXiv:1710.10196 (2017).
[8] Moura, Francisco. ”David Cope: A Lifetime Contribution to Artificial Intelligence and Music.” (2018)
[9] https://github.com/vishnubob/python-midi
[10] http://artsites.ucsc.edu/faculty/cope/experiments.html
[11] http://www.piano-midi.de
[12] http://mfiles.co.uk/
[13] https://magenta.tensorflow.org/datasets/groove
[14] http://www.generativemusic.com/scape.html
[15] http://www.generativemusic.com/bloom.html
[16] https://www.sonicvisualiser.org/
[17] https://www.aclweb.org/anthology/P16-1144.pdf
[18] Davis, Ernst, Morgenstern, Leora, and Ortiz, Charles. ”The Winograd Schema Challenge.” https://cs.nyu.edu/faculty/davise/papers/WinogradSchemas/WS.html