Artificial Intelligence in Music Composition

Mincer Alaeddine, Anthony Tannoury

Artificial Intelligence in Music Composition

Mincer Alaeddine[1][0000−0003−1951−4581] and Anthony Tannoury[2,3][0000−0002−0792−6434]

Antonine University, Baabda, Beirut, Lebanon anthony.tannoury@ua.edu.lb
https://ua.edu.lb/

Abstract. Technology has had a remarkable influence on music. As society advances technologically, the music industry does as well. An example that illustrates the use of technology in music is the use of artificial intelligence (AI) as a creative and inspiring tool. Music helps shape emotional responses, creates a rhythm, and comments on the action. It is often a very crucial element to any experience. However, music, like any form of art, is an extremely challenging field to tackle using AI. The amount of information in a musical structure can be overwhelmingly large. If we factor in the different and unpredictable nuances invoked by human imperfection and emotion, it becomes clear why, even though AI excels at handling large amounts of data, generating good music can be very challenging, especially when it comes to Jazz and similarly complex genres.

Keywords: Technology · Advancement · Music · Artificial Intelligence · Generation · Emotion · Imperfection · Genre · Training · Challenge · Prediction · Rhythm · Tempo · MIDI · Instruments · Patterns · Harmony · Melody · Caching

1 Introduction

In the past few years, AI has proven to be useful in the technical aspect of music. However, AI still lacks in the creative process, like music composition. The reason behind that becomes more evident as we start experimenting. The most common course of action for building AI applications is to train a model using a large dataset with enough data. That is known as batch training. That might be a good idea if music consisted of only a couple of genres, but that is not the case. Out of every genre, hundreds of subgenres emerge, and some pieces do not belong to any genre at all. Another challenge that music composition presents is the long-term interdependent structures [1]. Naturally, a music composition consists of repetition, and every note is somehow related to one or multiple musical notes in the same sequence. That makes it much harder for neural networks to learn without short-term memory [2]. The added value of this project is combining both online training and batch training to provide cross-genre predictions while maintaining accuracy. The focus will be on the online training as it will have the most impact on the output. While online training is very useful when working with flexible predictions and learning over time, it also comes at the cost of

speed and responsiveness, especially when it has to be done on the spot. Training models on the spot can be very resource-demanding, and this quickly becomes a problem the richer the input is, and from a UX perspective, for an application to be useable, it has to be highly responsive [3]. In the first chapter of this paper, we will go through the state of the art, shedding the light on other great projects in the field of AI for music composition, especially MIDI. Chapter 2 explains the added value of this project in detail by going over all the challenges imposed by music composition one by one, how we managed to solve those challenges, which challenges are still unsolved, and why.

2 Literature Review

2.1 Introduction

Music is, more often than not, classified by genres, which refers to the overall structure and character of particular musical composition. To create a track of a specific genre, we must meet some requirements like rhythm, tempo, key, and structure, to name a few. For example, when we hear an electronic dance music track or a disco track, we notice the kick on every beat [4]. While this might be an easy task to accomplish for humans, it is considered fairly complicated for AI. The purpose of this project is to provide an AI tool that takes a MIDI file from a user and generate multiple structurally similar pieces of music based on the data learned from said file.

2.2 Related Works

Google Magenta [5]The Google Magenta team has been working on blending AI with Music for a long time. Their projects Magenta and NSynth are great examples of how powerful AI can be when mixed with human creativity. The Magenta library contributed considerably to the execution of this project. Magenta Studio Allows:

- Generating drum beats for an existing melody
- Generating melodies from scratch
- Complete an already existing melody
- Interpolate between two samples

NSynth is still in beta. However, the main idea behind it is to synthesize a new sound using the acoustic properties of another sound, which tackles a different field in the world of music.

MuseNet [6] MuseNet is a general-purpose deep neural network (DNN) that can generate musical compositions with up to 10 instruments, combining styles from different epochs, artists, or bands. MuseNet, unlike other projects, was not trained through music knowledge, but instead discovered patterns of harmony, rhythm, and style by learning from hundreds of thousands of MIDI files.

Captain Plugins by Mixed In Key Mixed in Key is a digital music software company that has developed multiple products aiming to make the life of musicians easier. They are most recognized for creating Mixed in Key, a software that detects the key, BPM, and the energy of given songs. The company recently created a plugin suite called Captain Plugins. The suite focuses on generating melodies, and basslines, and drum patterns to a given chord progression.

This time with feeling: learning expressive musical performance [7] The authors of this journal discuss training a machine to generate music, focusing on the time and feel of the generated pieces. In simpler terms, the goal of their model is to produce human-like performances. Said model demonstrates generating music in the form of MIDI that successfully recreates the timing and velocity of a professional pianist. The music felt like an authentic performance by a musician. While this is great, it comes at the cost of the long-term structure, one of the problems tackled in this paper.

3 Problem Definition and Proposed Solutions

3.1 Introduction

Music composition, as we mentioned before, is a very complex field to tackle through AI because of the numerous challenges and problems that it presents. This chapter will go over the contribution of this paper to AI music composition, the challenges that we faced and how we tackled them.

3.2 Challenges and Solutions

Human Imperfection While computers excel at being perfect, humans tend to make mistakes sometimes. These unintentional mistakes are the soul of music and art in general. Without human imperfection, music will sound too generic and will lack emotion. In electronic music, artists tend to deliberately reproduce those imperfections to give the otherwise perfect song a bit of a human feel. For example, after drawing notes on a computer, artists tend to nudge the notes of the perfect tempo grid intentionally to reproduce the groovy feel of a real musician playing. The same goes for how hard musicians play the notes. Human imperfection remains one of the complex challenges of artificially generating music and is yet to be solved because imperfections are not the result of specific patterns and rules, but rather invoked through human emotion.

Cross-Genre Compatibility and Future Proofing Music is made of countless genres, and each genre of countless sub-genres, and every day, new genres and sub-genres are being created. When it comes to AI, this means one thing: Learning to generate music by associating an input with a specific genre is not the optimal solution as it requires thousands of hours of batch learning using

structured datasets, and all that training is rendered useless when the model encounters a new genre. Our model has been trained with an unstructured dataset of more than 13000 melodies from various genres. That, however, does not make our model future-proof, so in addition to that, we must integrate online training that will be done on-the-spot to learn the essence of the user's sample. That frees us of the genre issue and guarantees that the generated samples are always in the same style, tempo, and rhythm as the presented sample, nevertheless, this solution creates another problem, which is the amount of time and resources it takes to perform an online training on-the-spot.

Training On-The-Spot Training on the spot is both resource-dependent and time-consuming, thus, there is only so much that can be done to improve the responsiveness of such a resource-dependent process. The solution adopted in this project consists of using the cloud to perform all the heavy lifting, then rendering the result to the user using Server-Side Rendering or SSR for short. That helps us run our heavy processing on a platform that can adapt and scale accordingly, thus ridding us of the user's limited resources barrier. After multiple tests, I discovered that with the ability to retraining the model using the generated samples, the user will most likely try to retrain using a very similar sample, therefore, using very similar training data to generate new samples, thus, caching has been introduced to temporarily store the trained data, reducing the time it takes to perform a follow-up training by more than 50%.

Long-Term Structure in Music One of the core elements of a musical piece is structure. Songs are built of short sections, and every section includes patterns. For humans, it is natural to pick up on these patterns, thanks to our memory, and this is one of the main factors that make a song catchy and delightful to hear. However, when it comes to AI, this poses a challenge that requires a hierarchical decoder to solve. A hierarchical decoder is based on Long Short Term Memory networks or LSTM for short. LSTM is a type of RNN (recurrent neural network) that can learn order dependence in a given sequence [9], making it the perfect solution to the long-term structural problem.

Polyphonic Samples In most cases, creators will stumble upon a song that they like, search for the midi file for that song on the internet, and use it for online training. MIDI files on the internet often contain multiple instruments at once. This is called polyphonic music. However, generating a monophonic melody (a melody that consists of a single instrument) will not produce an accurate result if the data that is was processed is polyphonic especially if the notation of the instruments overlap. Therefore, it is necessary to extract the melody from the sample before proceeding with the training phase. This topic is still being researched and big music companies like Ableton are to this day struggling to get an accurate result by automatically extracting a monophonic melody from a polyphonic sample. For the sake of not re-inventing the wheel, we have used the Google Melody Extraction algorithm in this project.

4 Implementation[1]

4.1 Choice of Technology

For the front-end, the technology of choice was Nuxt.js. Nuxt is built on top of Vue.js, a JavaScript framework. Nuxt supports Server-Side Rendering out of the box and helps organize the project allowing it to scale as big as needed while keeping everything structured. These features combined with Vue's ease of use, make the process of creating extremely complex applications easier to manage and make sure everything is well organized. Additionally, Nuxt allows developing custom plugins that can be attached to any application seamlessly which helped us integrate Google's plugins and algorithms into our application. Furthermore, Nuxt integrates seamlessly with Google Cloud Services, which will be discussed in the Hosting section.

The batch learning process was done over Ubuntu using Anaconda and Python with the help of TensorFlow and the Magenta library. Anaconda makes it easier to manage long term projects by separating dependencies of every project in a virtual environment.

Google's Cloud Computing services were the hosting services of choice. The reason behind that choice is to benefit from the full potential of Server-Side-Rendering. Cloud hosting means that our project will not be limited to the user's resources, and it will scale with ease. Furthermore, this allows us to benefit from Google Cloud tools to optimize and improve in the future.

As for the final exported type, we chose to export MIDI files. Exporting as MP3 or any other codec adds unnecessary load to the server because it will force us to pre-load high-quality samples, which will then need to be rendered as audio, taking more time. Additionally, creators may often need to fine-tune the generated melody to fit their liking.

4.2 Implementation Process

Training the main model requires Magenta's development environment. Magenta was developed for Linux. In our case, we had to work with a Windows machine, therefore, we had to use WSL 2 (Windows Subsystem for Linux) to get started with the training without having to install a new OS. After downloading Magenta, it is recommended to create a new virtual python environment using Anaconda to avoid any unwanted conflicts and library issues with other projects. After finishing with the environment setup, we have to start building our dataset.

Curating a proper dataset is the most laborious and time-consuming part of every AI project. In our case, we had to collect thousands of midi files and sanitize them so we can train our AI model and avoid as much noise as possible. There were no specific requirements or features that decided whether a MIDI

[1] The project source code is available in the following GitHub repository: `https://github.com/minceralaeddine/ai-music-producer`

file is fit for the learning process or not. The reason behind that is that if we narrow all our MIDI files into 1 genre, style or tempo, we will be overfitting our model making it biased towards a specific genre, and it will no longer serve as a good base for other genres and styles. Therefore, any midi file could work as long as we balance out the genres. To accelerate the process of building the dataset, we used datasets with diverse genres from `https://archive.org/`, `https://bitmidi.com/` and the MAESTRO Dataset. Additionally, we used Mincer's own library of midi files to cover the electronic music genre, since he is an electronic music producer and already has a clean collection. We ended up with a dataset that consisted of more than 13000 midi files of diverse st yles, bpm, and scales.

Once the dataset is ready, we can begin the training process. The Magenta library makes it easy to train models using midi data. Magenta offers basic pre-trained models out of the box that we could have used, however, for our case, we had to train a custom model. Magenta's models are relatively basic and offer no flexibility, therefore, if the predictions were inaccurate we will need to revert to creating our custom model. We ended up creating four separate models using the magenta development environment: Two big models that require short on-the-spot training and two small models that require long on-the-spot training. One model of every batch is capable of generating a 16 bars melody, and the other is capable of generating a 4 bars melody. To start the training, we first need to convert our midi dataset to a TensorFlow-friendly type. Fortunately, the Magenta library makes that easy by providing a midi to note sequence converter, ready to be fed to our TensorFlow model. [8] To begin the training, we first need to convert our MIDI dataset into a format that we can work with using TensorFlow. We picked Magenta's NoteSequence [10] because it offers flexibility when it comes to exporting and manipulating MIDI files. Furthermore, note sequences contain information on how every note is played and where it is laid out in the song. All of this information will serve as input data. We could have easily parsed the MIDI file ourselves and optimized the attributes (like velocity and note length etc.) to create our own TensorFlow Record but that would be an unnecessary extra step since the conversion script is already there. Once the conversion is done, we can then initiate the training of our model using the generated sequences. The RNN used for this training is Magenta's Melody_RNN [12]. Melody_RNN is a neural network that generates monophonic melody predictions no matter how many voices it receives [13]. We could have used a polyphony RNN to generate polyphonic samples, but that would have taken a lot of time to get a decent accuracy as it involves harmony and chords [14]. That being said, the amount of time required to get an accurate polyphonic prediction will be directly dependent on the number of voices present in a chord. For example, training the model using a melody and strictly triads will take a lot less time than using a melody on top of 9^{th}, 11^{th}, or 13^{th} chords that are most commonly found in dominant chords in Jazz music.

Once everything is set up, we can use the trained model's checkpoint for the online training. The user will be asked to provide a MIDI sample of their choice.

Once the sample is uploaded, we need to convert it to a NoteSequence just like we previously did to train our model. Online training relies heavily on detecting MIDI attributes from the provided file. To avoid unexpected errors, we must predefine some default settings as constants to fall back to in case the MIDI analysis fails, like the default velocity and note length. After the conversion is completed, we then need to quantize the sequence to match the BPM grid and then proceed to split our note sequence into bars of the size that the model is going to generate (4 or 16). Quantizing helps detect the tempo accurately, however, this will cause pieces played by real musicians to lose their original groove and might drastically change the rhythm [15]. Since we already trained a base model, re-analyze the entire input sample will cause the training to take much longer than it should. We can use the base model as a checkpoint instead, and then analyze small parts of the sample, which is why we split it into 4 or 16 bar samples. To proceed, we encode our chunks using a variational autoencoder to ensure that there is enough variation to create a new sample and not just replicate the user's input, and then initiate the training. Variational Autoencoders (VAEs) are encoders that use complex mathematical encoders that can be applied to all sorts of data types [11]. Once the training is done, we generate 5 samples by calling the model.sample(1) command. Then we use the VAE again to decode that sample and using Tone.js, the user can listen to the results inside the browser. If the user was happy with one of the results and would like more of that sample, they could ask for new samples based on that result. The same algorithm will run again using the chosen sample as input data.

5 Experiments and Performance Analysis

5.1 Testing multiple genres and analysing the results

The model performs very well when the sample input is of a generic genre, like electronic music with an accuracy of 70% to 80% since human imperfection does not dictate the structure of such genres. However, when exposed to more complex genres, Middle Eastern music or Jazz, for example, we notice a drop in accuracy. The reason behind the confusion when it comes to Middle Eastern music is rather simple to understand but harder to incorporate into the network. Middle Eastern music, unlike western music, contains quarter tones, meanwhile, western music only consists of semi-tones and whole tones [16]. This causes confusion once our model runs into a quarter-tone because the way this is portrayed in MIDI is using microtones, which adds another layer of complexity to the network. For example, if the note that is being played is B semi-flat, it will be recorded as a B flat with 50 cents microtones in the MIDI sequence. Jazz, on the other hand, does not follow the same theory as other music genres and features unpredictable off-beat notes and complex chord progressions. Quantizing Jazz pieces will drastically alter the rhythm of the song so the predictions will be inaccurate. Leaving the piece unquantized, however, will lead to a failure in tempo detection therefore the predicted results will not respect the tempo of the original piece. To perform an in-depth analysis of our samples we loaded them into Ableton Live, a digital

audio workstation. If we compare the predicted sample from Avicii's Levels (see Fig. 1) with the original (see Fig. 2), we can see that the generated sample is in the same key and respects the rhythm of the original sample.

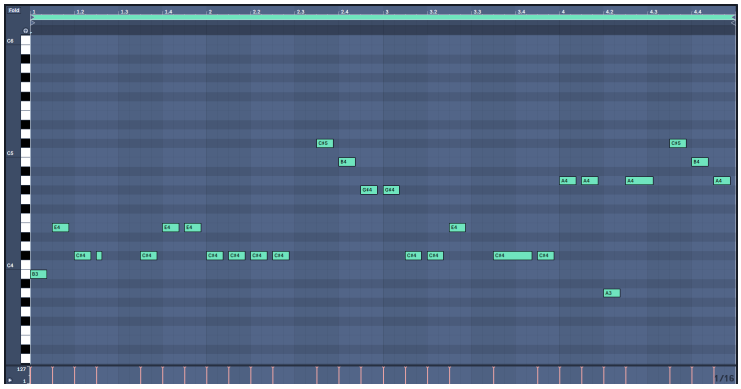

Fig. 1. Predicted Sample from Avicii-Levels. The vertical axis is the pitch, the horizontal axis is time, the green rectangles are notes, and the red bars at the bottom represent velocity.

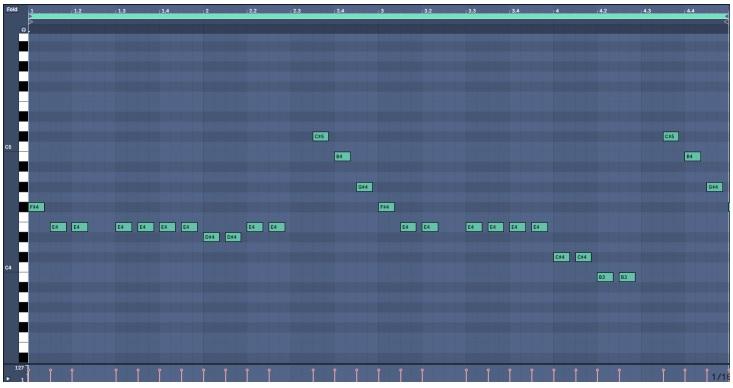

Fig. 2. Original Sample of Avicii-Levels

Dealing with an acoustic Jazz Guitar, however, produces a much less accurate result. If we compare the generated sample (see Fig. 3) with the original (see Fig. 4), we can see that even though the AI managed to detect the key and scale of the sample and extract the melody, the rhythm and feel is still off.

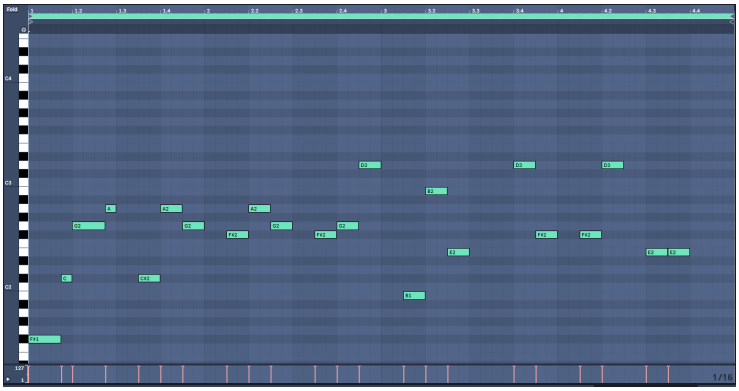

Fig. 3. Predicted Sample from a Jazz piano

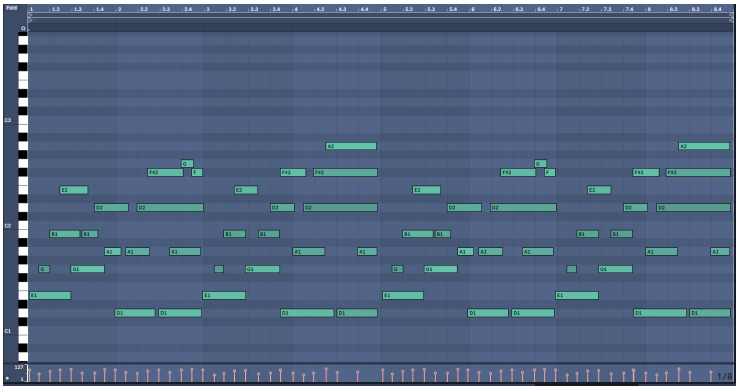

Fig. 4. Original Sample of a Jazz piano

We performed the analyses on five different genres, each using five samples. The results were as follows:

Table 1. Analysis of 25 predicted samples from 5 different genres.

GENRE	Rhythm Accuracy	Key Accuracy	Tempo Accuracy	Samples
Electronic Music	100%	80%	100%	5
Jazz	40%	80%	40%	5
Classical Music	80%	80%	100%	5
Middle Eastern Music	80%	20%	60%	5
Game Music	100%	80%	100%	5

5.2 Conclusion

The analysis and experimentation above showcase this model's capability of generating very accurate structures quickly until a certain level of structural complexity, then the accuracy starts to fall off. While it is true that we only used 25 samples in our analysis, it was more than enough to reveal the strong and weak points of this model. The weak points being complex genres like Jazz and Oriental Music, and the strong points being genres that follow strict music theory rules like classical music, electronic music, and game music.

6 Future Work

There is still a lot to improve on this approach of music generation through a combination of both online and batch training:

- Improving the accuracy for non-generic music played by instrumentalists
- Improving the accuracy of Jazz music generation by exposing the base model to more Jazz pieces so it can learn the theoretical exceptions
- Supporting middle eastern music by exposing the base model to more sequences with quarter tones and pitch bends
- Improving velocity detection for non-generic music genres

In the long, this approach can support a lot more complex features, to list a few:

- Using polyphony and harmony to create chords to complement the generated melody
- Generating drum beats for a given melody
- Using AI to identify weaknesses in a musical piece and suggesting improvements
- Providing high-level controls to the user allowing him to customize the predictions to his taste

7 Conclusion

AI will, without a doubt, have a major impact on the music industry and should not be looked at as a technology that will replace the creative process but as one that will develop it and open up more opportunities for creativity. Even though AI music composition faces some tough challenges, especially when it comes to making the result more humane, it's already doing a great job. Generating music using AI can help musicians overcome their creative barriers, it can automate the repetitive technical process that is inevitable in every music composition session, and it can expand the toolset of musicians allowing them to express themselves better by focusing on the creative process. To conclude, while humans excel at composing music and art, AI can still help humanity understand music even better and shed light on unexplored bits of the world of music composition.

List of Figures

References

1. M.R. Jones, *Dynamic pattern structure in music: Recent theory and research*. Perception & Psychophysics 41, 621634 (1987).
2. Analytics Vidhya, https://www.analyticsvidhya.com/blog/2020/01/how-to-perform-automatic-music-generation/. Last accessed 9 March 2021
3. UX Planet, https://uxplanet.org/why-fast-matters-a-lot-14c202e352f8. Last accessed 9 March 2021
4. Musical U, https://www.musical-u.com/learn/rhythm-tips-for-identifying-music-genres-by-ear/. Last accessed 7 Mar 2021
5. Tensorflow, https://magenta.tensorflow.org/studio. Last accessed 7 Mar 2021
6. MuseNet, https://openai.com/blog/musenet/. Last accessed 9 Mar 2021
7. Sageev Oore, Ian Simon Sander Dieleman, Douglas Eck, Karen Simonyan (2018) This time with feeling: learning expressive musical performance. Neural Computing and Applications (2020) 32:955967
8. Twilio, https://www.twilio.com/blog/training-a-neural-network-on-midi-music-data-with-magenta-and-python/. Last accessed 7 Mar 2021
9. Machine Learning Mastery, https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/. Last accessed 7 Mar 2021
10. GitHub Magenta, https://github.com/magenta/note-seq. Last accessed 9 Mar 2021
11. Jeremy Jordan, https://www.jeremyjordan.me/variational-autoencoders/. Last accessed 7 Mar 2021
12. GitHub Magenta, https://github.com/magenta/magenta/tree/master/magenta/models/melody_rnn. Last accessed 11 Mar 2021
13. Britannica Art, https://www.britannica.com/art/monophony. Last accessed 11 Mar 2021
14. Britannica Art, https://www.britannica.com/art/polyphony-music. Last accessed 11 Mar 2021
15. Midi.org Midi Quantization, https://www.midi.org/midi-articles/5-midi-quantization-tips-1. Last accessed 11 Mar 2021
16. Ghrab, A.: The Western Study of Intervals in "Arabic Music," from the Eighteenth Century to the Cairo Congress. The World of Music 47, no. 3 (2005): 55-79.