A.M.I.C.I.
AI Music Iterative Composer Interface

Bryan Healy
bhealy8@gatech.edu

1 INTRODUCTION

Recently in the field of AI generated music, much focus has been directed to the fidelity and performance of the music produced, attempting to reproduce long-term structure in musical composition and human nuance in performance. Advances in deep learning utilize large models to inherently learn music structure, human-like expression, and instrumental realism. However, use of these larger models have resulted in longer inference times, limiting their adaptability and interactivity in a real-time environment. Also, regardless of how much the models have improved, and how likely they are to create something beautiful or inspiring, there is always a note or even several bars that the musician wants to tweak. Sometimes, the generated music is overly chaotic and lacks coherency, other times it lacks variation and is unable to create surprising sequences.

This project will develop more responsive models and interfaces to interact with AI as an iterative creation tool for music exploration and generation, rather than just a black box. By lowering inference time of musical generation models, this project will improve interactivity and allow for quicker iterations.

Previous interactive systems for accompaniment generation rely on hard-coded rules to determine timing and sound of generated audio. This leads to the musician needing to experiment with the system to figure out how to work their music around the AI's. For example, most musical accompaniment models will wait a certain amount of time or beats before clipping the musician's performance as an input sequence. By letting the musician control the start and end of their sequence, they will never be cut off or left waiting. This project will create an interface that puts the control back into the hands (and feet) of the musicians, opening them up to new possibilities, and making AI more collaborative.

2 PREVIOUS WORK

2.1 Music Generation

Of the many music-generating deep learning models, this section will focus on those used for improvisational accompaniment and call-and-response, as they are more adaptable in a real-time application than models which compose music offline.

Early work with the Shimon Robot used Hidden Markov Models to determine what note to play next based on the pitch and rhythmic values of the input melody (Nikolaidis et al, 2010). The use of Genetic Algorithms to mutate and cross-breed a given musical sequence was used in the robot percussionist Haille (Weinberg et al, 2020). Similarly, Variational Autoencoders have been used to model a latent space of plausible melodies, which were then used to produce novel melodies through interpolation of two given melodies (Roberts et al, 2018).

More recent work by Google Magenta has made improvements in long-term song structure using Lookback and Attention mechanisms built upon LSTM cells (Waite et al, 2016). Both models have been used in various applications including one that used the LSTM-based model to synchronize loops of generated music with recorded human-created loops (Castro, 2019). Later work by Magenta includes Music Transformer, which builds upon the previous models' feature representations, but uses Transformers with a modified attention mechanism to represent "relative positional information" with reduced memory requirements. Music Transformer is capable of generating minute-long compositions, continuations of given motifs, and melody accompaniments (Huang et al, 2018).

Other approaches have used Deep Reinforcement Learning to learn a representation of musical structure and a "policy to generate a musical note" based on the input musical sequence (Jiang et al, 2020).

2.2 Musical Expression

Previously, synthesizers have been used to generate sound from a given MIDI sequence. While they provide "detailed expressive controls," they lack the nuances of playing a physical instrument (Wu et al, 2021). Models that synthesize musical audio through direct waveform construction, such as WaveNet, have proved to create realistic sounding audio. However, they take a significant

amount of time to run inference due to the creation of thousands of samples per second, processed with a sequential attention mechanism (Oord et al, 2016).

Google Magenta has created several deep neural networks to model the timbre of various live-recorded instruments. For example, Neural Audio Synthesizer (NSynth) was created to synthesize performances of a particular instrument using a WaveNet-style autoencoder (Engel et al, 2017). The model can even interpolate between instruments to create interesting combinations. A later model, GANSynth, uses Generative Adversarial Networks to perform similar functions given the notes from a MIDI file (Brock et al, 2018). Both are trained using a large dataset also called NSynth containing high-quality recordings of single notes played by 1006 instruments (Engel et al, 2017).

To demonstrate the NSynth model, the Magenta team created NSynth Super, an interface that allowed playback from a MIDI instrument using precomputed combinations of instrument sounds. A touchpad was used to select the amount each of four instruments contributed to the produced sound. Additionally, the interface used six fine-tuning potentiometers to adjust the position, attack, decay, sustain , release, and volume of the produced sound (NSynth Super, 2018).

Recently, Magenta created the Differentiable Digital Signal Processing (DDSP) model. This deep learning model applies efficient and interpretable "classic signal processing elements" to the problem of audio synthesis. While producing high-fidelity audio, DDSP also allows for independent manipulation of several components of the generated audio such as pitch and loudness (Engel et al, 2020). Another version of the model was also created to use MIDI sequences as input (Wu et al, 2021).

2.3 Optimization for Real-Time Model Inference

Advancements in processing power and chip design have enabled the use of larger deep learning models for real-time applications. For example, Tensor Processing Units (TPUs) have been created to "accelerate the inference phase of neural networks" by specializing in the matrix mathematics used in these calculations (Jouppi et al, 2017). Recently, TPUs have been applied to the embedded system industry with the creation of Google Coral, which provides TPU accelerators integrated into small platforms similar to RaspberryPi (Dev Board n.d.).

Along with more powerful hardware and improvements to deep learning models to lower inference time, there have been many tools and techniques developed to increase throughput. Quantization of model parameters from 32-bit floating point values to lower precision 8-bit fixed point values results in higher inference throughput, less memory, and less energy usage with relatively small decreases in accuracy (Gholami et al, 2021). Similarly, model pruning aims to reduce the size, and consequently the number of calculations needed for inference, by removing neurons which have a relatively small effect on the output of the model (Han et al, 2015). TensorFlow Lite is a library built by Google which optimizes deep learning models for inference time and memory usage to improve its usability on a mobile or embedded device (Performance best practices, 2022).

2.4 Real-Time Playback Synchronization

Work has been done to synchronize generated music with a human performer by using various audiovisual cues. Some have used audio cues such as digital metronomes that the human musician used to keep in sync with the generated music (Castro, 2019). Similarly, a system of several synchronized, physical metronomes with conductor batons have been used to synchronize multiple musicians playing concurrently in different time signatures (Godfried-Willem, 1997). Godfried-Willem also proposes several physical interfaces for musicians to indicate the beat to a music sequencer, including a foot pedal and MIDI-baton (Godfried-Willem, 1998). The Shimon Robot has also used similar techniques, tracking infrared lights attached to a percussionist's mallets as cues for beat synchronization (Bretan et al, 2012).

2.4.1 Beat Detection

Other work has been done to derive the tempo and time signature from the musician's instrument itself, removing the need for any additional sensors or controls. In their book "Robotic Musicianship", Weinberg et al describe their method of beat tracking developed for the Shimi Robot, which derived the tempo from a combination of low-level spectral domain features obtained from a Short Time Fourier Transform (Weinberg et al, 2020).

3 PROPOSED WORK

3.1 Semester 1 - Accompaniment Generation (MIDI Sequence-to-Sequence)

The first semester will be focused on selecting a music generation model. With the goal of creating an interactive accompaniment generation system, low latency inference is critical to a responsive experience that is quick to iterate with.

An investigation of the various classes of deep learning models (including Transformer-based, LSTM-based, etc.) will determine their current real-time improvisational capabilities. The three models hypothesized to be the most adaptable will be implemented and tested for various real-time performance measures, including inference speed and memory usage, using the TensorFlow Lite benchmarking tools. The performance evaluations will be done before the models are trained, using only randomly initialized parameters, as the actual accuracy of the models is irrelevant at this stage. The highest performing model will then be trained using self-supervised learning, using segments of MIDI tracks as input and the corresponding segment of a complementary track as ground truth. Instead of quantizing the model parameters after training, they will be set to use 8-bit fixed point values during training. Next, the model will be further optimized for inference time by pruning redundant and irrelevant neurons and analysis will be performed using TF Lite benchmarking. The testing accuracy of each optimization iteration will also be observed and the model achieving the best inference performance while also maintaining an acceptable level of accuracy will be chosen for the remainder of the project. A public domain grand piano sampling library will be used to synthesize audio from the generated MIDI sequences (Salamander GrandPiano, 2022).

A performance analysis on the Coral Dev Board will determine the viability of using it to host the music generation model. This analysis will inform the design of the rest of the system. If the board is able to run the model quick enough for an interactive experience, the remaining system utilization will determine upper bounds for concurrent processes and models loaded into memory.

3.2 Semester 2 - Iterative Interaction

An embedded system based around the Google Coral Dev board will be built to give the musician control of the AI models. A touchscreen will be used to provide a customizable interface for the musician. Several multi-purpose dials

will also be used for quick, tactile control of the various parameters of the chosen generative model, and later features such as instrument selection. A foot pedal will also be used to control the start and end of an input sequence recording.

After recording a sequence, the generative model will produce an accompaniment and display it. The musician will then be able to listen to the accompaniment played along with the original. Hearing a portion of the generated sequence that they like, they will be able to select that portion to be used as additional conditioning. Based on the selected portion of the sequence, several variations will be generated. If one of the newly generated accompaniments is pleasing to the musician, they will be able to select it to replace the original generation. The musician can continue this iterative process of selecting the portions they enjoy, and re-generating the rest of the sequence. When finished, the musician will then be able to save the recording and accompaniment to a MIDI file.

Design of the iterative model will explore the possibility of extending the chosen generation model to the problem of "in-painting" or using a separate model instead.

3.3 Semester 3 - Enhanced Expression

Improvements will be made to the interface which allow the musician to compose an entire song by combining multiple recordings and accompaniment generations.

Several more realistic instrument sounds, produced by a model such as GANSynth or MIDI-DDSP, will be added as selectable voicings for each track. The musician will be able to modify available parameters of the chosen voicing model during playback, using the configurable physical controls. This will allow for unique performances with dynamic changes throughout the duration of the song, which can be recorded to a file.

If it was determined that the Coral board was not powerful enough to run the generation models or both the generation model and the voicing model, then a server will be configured to run the generation model, while the Coral board runs only the voicing model and acts as an interface to the generative models on the server.

4 DELIVERABLES

4.1 Semester 1

1. Weekly status reports, detailing progress the past week, the challenges encountered, and any revised expectations for the project
2. Paper describing:
    a. the different music generation architectures,
    b. why the chosen three were believed to be more adaptable to a real-time environment
    c. results of their inference performance analysis
3. Code written to perform inference performance analysis using TF Lite benchmarking
4. Code which:
    a. integrates the MIDI dataset
    b. trains the chosen model using quantized parameters
    c. prunes the model
    d. tests the model's retained accuracy post-optimization
5. Report describing:
    a. the chosen model
    b. how and why it was optimized for inference speed
    c. results of optimization and accuracy back-testing
6. Paper describing:
    a. the goal of the semester's work
    b. results of experiments and how they informed what was built
    c. the future design of the system given what was learned
    d. the work to be done in Semester 2
7. Blog post embedded with video demo of the generative model running on the Coral Dev board (if possible, otherwise on PC), along with commentary and example generations

4.2 Semester 2

1. Weekly status reports, detailing progress the past week, the challenges encountered, and any revised expectations for the project
2. Iterative model to produce variations
3. Paper describing iterative generation model
4. Software process to run generative and iterative models

5. Coral hardware interface: MIDI input, touchscreen, pedal input, dials and necessary firmware
6. Software to:
    a. record input MIDI sequence
    b. interact with model process
    c. display recording and generated sequence
    d. allow selection of sequence section
    e. interface for iteration
    f. playback, and saving to a file
7. Video describing interface design
8. Paper describing:
    a. the goal of the semester's work
    b. results of experiments and how they informed what was built
    c. the future design of the system given what was learned
    d. the work to be done in Semester 3
9. Blog post embedded with video demo of the iterative generation model running on the Coral Dev board (if possible, otherwise on PC), along with commentary and example generations

4.3 Semester 3

1. Weekly status reports, detailing progress the past week, the challenges encountered, and any revised expectations for the project
2. Code updates to improve interface as described
3. Video demonstrating interface upgrades
4. Code to generate dynamic voicings
5. Video demonstrating dynamic voicings
6. Server setup if Coral alone is not powerful enough, along with supporting program and networking code
7. Final paper describing:
    a. the goal of the project
    b. results of experiments and how they informed what was built
    c. the design of the system given what was learned
    d. Any future work to be done
8. Blog post embedded with demo video of the song creation process, using all features built, along with commentary and example generations

5 SEMESTER 1 TASK LIST

Week #	Task #	Task Description	Estimated Time (Hours)	Member Responsible		
1		Research generative models and choose 3	15	Bryan		
1		Weekly Status Report	0.5	Bryan		
2		Implementation and testing model 1	10	Bryan		
2		Weekly Status Report	0.5	Bryan		
3		Implementation and testing model 2	10	Bryan	Total Hours	**188**
3		Weekly Status Report	0.5	Bryan		
4		Implementation and testing model 3	10	Bryan	Bryan	188
4		Weekly Status Report	0.5	Bryan		
5		Paper describing performance analysis and model comparison	10	Bryan		
5		Weekly Status Report	0.5	Bryan		
INTERMEDIATE MILESTONE 1 DUE						
6		Model quantization	10	Bryan		
6		Weekly Status Report	0.5	Bryan		
7		Dataset integration	10	Bryan		
7		Weekly Status Report	0.5	Bryan		
8		Model training	15	Bryan		
8		Weekly Status Report	0.5	Bryan		
9		Model training	15	Bryan		
9		Weekly Status Report	0.5	Bryan		
10		Model pruning and accuracy back-testing	15	Bryan		
10		Weekly Status Report	0.5	Bryan		
11		Report describing model choice and optimization results	10	Bryan		

11		Weekly Status Report	0.5	Bryan		
INTERMEDIATE MILESTONE 2 DUE						
12		Coral Dev Board bringup and testing	10	Bryan		
12		Weekly Status Report	0.5	Bryan		
13		Testing model on Coral	5	Bryan		
13		Coral performance analysis	2	Bryan		
13		Weekly Status Report	0.5	Bryan		
14		System Design	10	Bryan		
14		Weekly Status Report	0.5	Bryan		
15		Semester Paper	15	Bryan		
15		Weekly Status Report	0.5	Bryan		
16		Video Demo	5	Bryan		
16		Blog post	3	Bryan		
16		Weekly Status Report	0.5	Bryan		
FINAL PROJECT DUE						

6 REFERENCES

1. Brock, A., Donahue, J., & Simonyan, K. (2018). Large scale GAN training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096.
2. Bretan, M., M. Cicconet, R. Nikolaidis, & G. Weinberg (2012). Developing and composing for a robotic musician. In Proceedings of International Computer Music Conference (ICMC'12). Ljubljana, Slovenia.
3. Castro, P. S. (2019). Performing structured improvisations with pre-trained deep learning models. arXiv preprint arXiv:1904.13285.
4. Dev Board. (n.d.). Coral. https://coral.ai/products/dev-board/
5. Engel, J., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D., & Simonyan, K. (2017, July). Neural audio synthesis of musical notes with wavenet autoencoders. In International Conference on Machine Learning (pp. 1068-1077). PMLR.
6. Engel, J., Hantrakul, L., Gu, C., & Roberts, A. (2020). DDSP: Differentiable digital signal processing. arXiv preprint arXiv:2001.04643.

7. G. (2018). GitHub - googlecreativelab/open-nsynth-super: Open NSynth Super is an experimental physical interface for the NSynth algorithm. NSynth Super. https://github.com/googlecreativelab/open-nsynth-super
8. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. (2021). A survey of quantization methods for efficient neural network inference. arXiv preprint arXiv:2103.13630.
9. Godfried-Willem. (1996). Polymetronome: A musical conductor robot. Logos Foundation. https://www.logosfoundation.org/instrum_gwr/metronome/polymetronome.html
10. Godfried-Willem. (1997). Synchronizing sequencers with performers on stage. Logos Foundation. https://www.logosfoundation.org/kursus/2320.html
11. Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.
12. Huang, C. Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., ... & Eck, D. (2018). Music transformer. arXiv preprint arXiv:1809.04281.
13. Jiang, N., Jin, S., Duan, Z., & Zhang, C. (2020). RL-Duet: Online Music Accompaniment Generation Using Deep Reinforcement Learning. Proceedings of the AAAI Conference on Artificial Intelligence, 34(01), 710–718. https://doi.org/10.1609/aaai.v34i01.5413
14. Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., ... & Yoon, D. H. (2017, June). In-datacenter performance analysis of a tensor processing unit. In Proceedings of the 44th annual international symposium on computer architecture (pp. 1-12).
15. Nikolaidis, Ryan, and Gil Weinberg. (2010). Playing with the masters: A model for improvisatory musical interaction between robots and humans. In 2010 IEEE, RO-MAN, 712–717. IEEE
16. Oord, A. V. D., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... & Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499.
17. Performance best practices | TensorFlow Lite. (2022). TensorFlow. https://www.tensorflow.org/lite/performance/best_practices

18. Roberts, A., Engel, J., Raffel, C., Hawthorne, C., & Eck, D. (2018, July). A hierarchical latent vector model for learning long-term structure in music. In International conference on machine learning (pp. 4364-4373). PMLR.
19. Salamander GrandPiano. (2022, March 4). Yam. https://rytmenpinne.wordpress.com/sounds-and-such/salamander-grand piano/
20. Waite, E.; Eck, D.; Roberts, A.; and Abolafia, D. (2016). Project magenta: Generating long-term structure in songs and stories
21. Weinberg, G., Bretan, M., Hoffman, G., & Driscoll, S. (2020). Robotic Musicianship: Embodied Artificial Creativity and Mechatronic Musical Expression (Automation, Collaboration, & E-Services, 8) (1st ed. 2020 ed.). Springer.
22. Wu, Y., Manilow, E., Deng, Y., Swavely, R., Kastner, K., Cooijmans, T., ... & Engel, J. (2021). MIDI-DDSP: Detailed control of musical performance via hierarchical modeling. arXiv preprint arXiv:2112.09312.