BeatQraft: Quantum Generative Modeling for Novel Beat Creation

Rohan S. Kumar,[1] Ramachandran S. Srikanthan,[2] Neha Suresh,[2] Kshitij Duraphe,[3] and Stefanie Muroya Lei[4]

[1]*University of Chicago, Chicago IL*
[2]*North Carolina State University, Raleigh NC*
[3]*Boston University, Boston MA*
[4]*Institute of Science and Technology Austria, Austria*

I. Motivation

Quantum Generative Models have been shown to have numerous applications, in fields from drug discovery [1] to finance [2]. Furthermore, quantum generative models are known to be more expressive than their classical counterparts [3]. In parallel, the applications of quantum computing to music has been a popular topic at many quantum computing hackathons and quantum challenges. BeatQraft attempts to fuse these two popular areas of investigation. Namely, BeatQraft seeks to leverage the enhanced expressive and generative capabilities of quantum generative models to aid musicians and composers in the synthesis of novel beats.

BeatQraft's target market is comprised of amateur and professional composers, a group that regularly experiences writer's block and generative inertia. BeatQraft seeks to solve this problem for its users by leveraging quantum generative models to inspire a composer's next piece or soundscape. Concretely, BeatQraft uses a Quantum Circuit Born Machine (QCBM) [4] to learn a target distribution of valid beats. After training, the QCBM encodes not only the beats it trained on, but numerous other beats that emerge from the underlying features of the training set. By sampling from the output distribution of the trained QCBM, BeatQraft is able to generate potentially novel beats to lift a musician out of writer's block. QCBMs have already been shown to have a strong capacity for generalization [5], making them the ideal architecture for BeatQraft. Furthermore, the combinatorial nature of this problem means that there is enormous potential for BeatQraft to generate entirely novel music as practical quantum devices scale.

II. Encoding and Training

We encode beats to bitstrings by assigning a bit to each semiquaver in a bar. Specifically, assuming a 4/4 time signature, we split the bar into 8 subdivisions, represented as an 8-bit bitstring. A '1' in index k of the bitstring indicates that there is a 'hit' on subdivision k of the bar.

We use three different training sets corresponding to hi-hat rhythms, snare rhythms, and bass drum rhythms. At iQuHack, we constructed these training sets by transcribing popular songs from Spotify by hand, and by permuting the resultant bitstrings. However, in the future/at scale, we anticipate that a beat scraping protocol will be highly useful in producing these training sets.

We train three QCBMs using each of the training sets. These QCBM trainings were done using the IBM QASM simulator due to the constraints in the size and viable circuit depth of IBMQ Oslo and Nairobi. When constructing a new beat, BeatQraft will sample from each of the three trained QCBMs (corresponding to the hi-hat, snare, and bass) and superpose the resultant bitstrings to create a beat. BeatQraft samples from these distributions using multiple instances of random circuit sampling on IBMQ-Oslo using Covalent.

All QCBMs were trained with the ADAM optimizer using a learning rate of $\alpha = 0.2$ and a step size of $\eta = 0.1$. These parameters were inspired by the training of a similar QCBM used in [6]. The QCBMs were trained for 200 iterations with 300 shots per iteration. These hyper-parameters were chosen due to a combination of feasibility and time constraints. Given access to more classical or quantum resources, BeatQraft would be able to undergo even more robust training.

III. Presentation

We developed a website in Python using Flask as a framework for the backend. Our website draws from open-source code and copyright-free images. We integrated our beat generation algorithm into the website's backend, enabling users to generate beats at the push of a button. We wrote a python script to transform the bitstrings outputted by BeatQraft into a MIDI file that the user can listen to on our webiste. The user can currently generate as many beats as they want on our pre-trained QCBMs. When

presenting a beat on our website, we loop the beat four times to give the artist an indication of how their beat will sound within a larger project.

IV. BeatQraft at Scale

As hinted at throughout our report, BeatQraft has tremendous promise as its available resources grow. First, given access to larger quantum devices or more powerful simulators, BeatQraft could add more layers and represent more sophisticated beats. In addition, with more qubits, BeatQraft could feasibly train on *concatenations* of different layers of the training set, enabling it to capture correlations between different components of the beats. For example, if BeatQraft trained on concatenations of the hi-hat and the snare drum layers, it would be able to learn more deeply about the connections between these components in a training set. Second, with a robust beat-scraping protocol, BeatQraft could have access to a far larger training set, enabling it to exhibit further expressivity and generalization. Lastly, generative models are naturally more capable of expressing their capabilities at scale due to the inherent limitations of small sample spaces.

V. Conclusions

The creation of good music has been a coveted talent for centuries. Making a good song is quite difficult as one must know all the notes and their arrangements to make a good song. We have challenged talented individuals by successfully writing an algorithm that creates music supported by powerful new technology. Initial iterations of this project have tried to generate holistic soundscapes with pitch and rhythm. We would like to add rhythm and melody to the generated chords and explore more complex stochastic processes for generating music. We would also like music to be parsed by genre (example: pop music filter). We were content with the overall outcome and scalability of our project.

[1] Junde Li, Rasit O Topaloglu, and Swaroop Ghosh, "Quantum generative models for small molecule drug discovery," IEEE transactions on quantum engineering **2**, 1–8 (2021).
[2] Dylan Herman, Cody Googin, Xiaoyuan Liu, Alexey Galda, Ilya Safro, Yue Sun, Marco Pistoia, and Yuri Alexeev, "A survey of quantum computing for finance," arXiv preprint arXiv:2201.02773 (2022).
[3] Xun Gao, Eric R Anschuetz, Sheng-Tao Wang, J Ignacio Cirac, and Mikhail D Lukin, "Enhancing generative models via quantum correlations," Physical Review X **12**, 021037 (2022).
[4] Jin-Guo Liu and Lei Wang, "Differentiable learning of quantum circuit born machines," Physical Review A **98**, 062324 (2018).
[5] Kaitlin Gili, Mohamed Hibat-Allah, Marta Mauri, Chris Ballance, and Alejandro Perdomo-Ortiz, "Do quantum circuit born machines generalize?" arXiv preprint arXiv:2207.13645 (2022).
[6] Kaitlin Gili, Mykolas Sveistrys, and Chris Ballance, "Introducing non-linearity into quantum generative models," arXiv preprint arXiv:2205.14506 (2022).