DeepDrummer : Generating Drum Loops using Deep Learning and a Human in the Loop

Guillaume Alain[1][2], Maxime Chevalier-Boisvert[1][2], Frédéric Osterrath[2], and Rémi Piché-Taillefer[2]

[1] co-first authors guillaume.alain.umontreal@gmail.com
[2] Mila - Quebec Institute of Learning Algorithms

Abstract. DeepDrummer is a drum loop generation tool that uses active learning to learn the preferences (or current artistic intentions) of a human user from a small number of interactions. The principal goal of this tool is to enable an efficient exploration of new musical ideas. We train a deep neural network classifier on audio data and show how it can be used as the core component of a system that generates drum loops. We aim to build a system that can converge to meaningful results even with a limited number of interactions with the user. This property enables our method to be used from a cold start situation (no pre-existing dataset), or starting from a collection of audio samples provided by the user. In a proof of concept study with 25 participants, we empirically demonstrate that DeepDrummer is able to converge towards the preference of our subjects.

Keywords: drum loop generation, human in the loop, deep learning, active learning

1 Introduction

Modern day music production typically relies on a variety of software tools. These tools can help make music creation more accessible by streamlining an artist's workflow. Machine learning offers the promise that we may be able to build tools that not only automate certain tasks, but also have a certain level of understanding of an artist's musical taste and vision. In this work, we address the challenge of training a deep learning model to approximate the musical tastes of a human user with limited feedback for the purpose of creating an interactive tool that generates drum loops. This approach has a wide range of potential applications, one of which is to help creators find inspiration. We encourage the reader to view this from the perspective that this is a new method, and that it is not a fully-fledged music production system yet.

We present a system that we call DeepDrummer, which is composed of three main components: the interface, the critic and the generator. Users are faced with an interface where they sequentially listen to individual drum loops and rate them by clicking on either *like* or *dislike* buttons. For each user, we train a deep neural network from scratch to predict their ratings based on an audio signal.

This neural network constitutes the *critic* that judges the quality of drum loops produced by the *generator*. The generator is a function that outputs random grid sequencer patterns with 16 timesteps during which 4 randomly-selected drum sounds can be triggered. We choose a very basic generator that does not have any trainable parameters, and constitutes a source of patterns that has few priors on musical structure. Combined together, the feedback from the critic can serve as a powerful filter for the output of the generator. As a result, the interface will present only the most relevant drum loops to the user for rating.

DeepDrummer learns interactively while gathering data from a limited number of human interactions. This is somewhat contrary to the common wisdom in deep learning that training a useful deep neural network necessarily requires hundreds of thousands, or even millions of data points. Since human preferences are subjective by nature and may change as they are being measured, creating a perfect model of a person's musical preferences is a non-goal. We are instead interested in creating a useful filter that is utilized to quickly explore musical ideas in a way that is artistically useful.

The core contributions of this paper are the following:

- We present DeepDrummer, a system that features a novel approach to combine deep learning models and active learning to generate drum loops based on limited human interactions.
- We run an experiment with 25 participants to empirically measure the performance of such a system, and we show that gains are made within 80 interactions (p-value 0.00013).
- We publish DeepDrummer as an open source software, as well as all the data we collected during our experiment, which includes 3500 generated drum loops and associated user ratings.

To accompany this paper, a video showcasing DeepDrummer being interactively trained is available on YouTube[3], and the source code is available on GitHub[4]. The experimental data collected is available through a shared Google Drive link[5]. The total size of the download is 2.5GB.

2 Related Work

The problem of music generation and creativity has attracted much attention throughout community in recent years, in part because progress in deep learning opens many interesting research directions. Many methods frame music generation as a sequence modeling problem (Oord et al., 2016; Manzelli, Thakkar, Siahkamari, & Kulis, 2018; Dhariwal et al., n.d.; Gillick, Roberts, Engel, Eck,

[3] https://youtu.be/EPKsUf5YBeM
[4] https://github.com/mila-iqia/DeepDrummer
[5] https://bit.ly/363cBdj

& Bamman, 2019; Vogl & Knees, 2016; Lattner & Grachten, 2019). The general assumption is that the right probabilistic model will capture a manifold of desirable music to draw from.

Another popular approach to generate any kind of artistic content is that of Generative Adversarial Networks (GAN) (Goodfellow et al., 2014). The core idea is that a distribution can be learned by combining two neural networks, a generator and a discriminator, in a configuration that represents the equilibrium of a game where the two networks are trying to achieve opposite goals. This differs from the classic approach of loss minimization. There have been a variety of applications to the concept in the domain in music generation (Dong, Hsiao, Yang, & Yang, 2018; Engel et al., 2019; Donahue, McAuley, & Puckette, 2018; Kumar et al., 2019; Vogl, 2018). DeepDrummer draws some inspiration from the concept of generator and discriminator networks of GANs, but there is no adversarial training in DeepDrummer. Moreover, there is no reference dataset that the generator seeks to reproduce. We use the term *critic model* to highlight the fact that this is not a GAN discriminator.

DeepDrummer fits in the trend of computer music generation by integration of deep learning in the composition process, but it does not fall in the sequence-modeling approach. Instead, one of the novel contributions of DeepDrummer is to have a generator-critic framework in which the computational heavy lifting is done exclusively by the *critic* model. Our work shares similarities with the work of (Jaques, Gu, Turner, & Eck, 2017) in which an agent in a reinforcement learning environment produce music using a sequential model, though their approach rewards adherence to music theory through an external set of rules.

DeepDrummer receives feedback from users through a simple interface that involves a binary choice of *like* or *dislike*. This kind of interaction is reminiscent of the work of (Eric, Freitas, & Ghosh, 2008) in which the authors use Gaussian processes in order to navigate a complex space of parameters based on minimal user feedback. Their motivating use case involves graphics rendering in which the meaning of the exposed parameters does not translate intuitively into the visual output of the system. Humans are very good at judging the visual output of such systems. It then makes sense for a system to capitalize on this ability (this is also discussed in (Wilson, Fern, & Tadepalli, 2012)). Incidentally, this concept from (Eric et al., 2008) has been implemented in a musical setting by (Huang et al., 2014), in which the authors developed a platform that applies active learning to learn higher-level intuitive synthesizer knobs by querying users about perceived sound quality. One of the notable differences in the case of DeepDrummer compared with the work of (Eric et al., 2008; Huang et al., 2014) is that we are not solving an optimization problem with the goal of reaching the global maximum but rather to suggest many good candidates.

In active learning, a system queries the user strategically, so as to gain as much useful information as possible while minimizing the number of interactions. In a context where a classifier is trained, the system will often focus on the decision boundary where all the ambiguous data points are found. In the case of DeepDrummer, a similar phenomenon occurs, though maximal learning opportunities instead arise when DeepDrummer produces erroneous drum loops that it confidently feels that the user will *like*. Much like in the case of active learning, the training set grows over time, but contains only data points that are highly informative in order to correct misconceptions the critic has about the user. See (Christiano et al., 2017) for a recent example of reinforcement learning with minimal human interactions.

Finally, there is also an element of commonality between DeepDrummer and music recommendation systems (MRS) that are based on audio similarity between songs (Bogdanov et al., 2011; Lops, De Gemmis, & Semeraro, 2011; Wang & Wang, 2014; Zangerle & Pichl, 2018). DeepDrummer's critic model takes audio data in order to make a prediction of the user's probability of *liking* it. This allows DeepDrummer to generalize across the various sounds found in drum loops.

3 Model and Framework

3.1 Overview

The basic pipeline for DeepDrummer is shown in Figure 1 and consists of the following components:

- a random drum pattern *generator* to propose initial grid sequencer patterns;
- a library of one-shot audio samples including the kind of sounds usually found in a drum kit;
- a function that renders grid sequencer patterns and a list of associated drum samples (one per grid row) into an audio waveform;
- a neural network classifier *critic* to determine the desirability of the drum loop audio by outputting a value in the interval $[0, 1]$, which is a prediction of the odds of the user *liking* that drum loop;
- a web interface to present the human user with drum loops and get feedback as *like*/*dislike* ratings.

In the context of the experiments described in this paper, for simplicity, we are always working with 16-step patterns with 4 instrument tracks (4 one-shot samples). These form one bar, or two seconds of audio at 120bpm, which is rendered into monophonic audio at 44.1kHz. We draw from a varied collection of 340 one-shot samples.

3.2 Network Architecture of the Critic

A few different network architectures were considered for the critic, but the architecture we ultimately chose is one that uses Mel-Frequency Cepstral Coefficients (MFCCs) (Davis & Mermelstein, 1980) as input features. The network uses a stack of 4 layers of 2D convolutions (with 64, 64, 64, then 8 channels), each with leaky ReLU activation and batch normalization to downsample the MFCC features. The dimensions of the hidden representations is reduced each

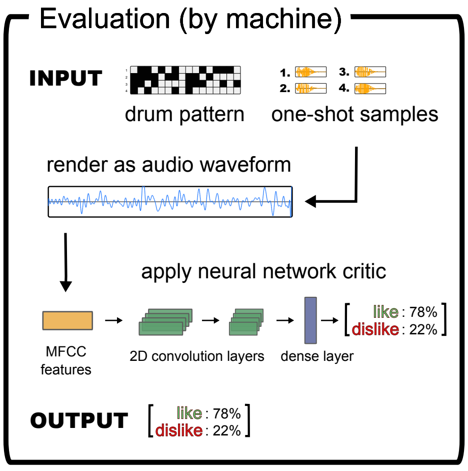

Fig. 1: Sketch of the core components of DeepDrummer.

time by using convolution kernels of size $(4, 8)$ and strides $(2, 4)$. This is then followed by one dense layer of 128 units (with leaky ReLU). A final mapping down to two units culminates in a softmax layer that estimates the probability that the user will *like* or *dislike* a given input. The network is relatively small at just 291K learnable parameters. This is intentional, as small networks tend to be easier to train with limited data. Dropout and weight decay regularization are used to prevent overfitting.

The neural network is trained in a supervised fashion based on the feedback given by the user. The fact that it takes an audio input, instead of symbols, means that it can learn about what our user wants from the perspective of the sound itself instead of just from the mapping of the instruments. This makes it possible for the classifier to generalize between variants of kicks (or hi-hats, or any other sounds). We did not manually sort instruments into categories, nor give DeepDrummer any idea that certain sounds should play a specific role in drum loops. An upside of this approach is that it can lead to creative use of the sounds in the collection, and it is easy to add to the existing collection by simply copying audio files to a single directory.

The critic $f_\omega(x) \in [0, 1]$ is initialized randomly for every user, where ω are the user-specific parameters and x a drum loop. The training set is constituted of all the previous drum loops rated by the user, and it grows every time we get a new rating. By minimizing the cross-entropy loss of the critic during training, a model that generalizes well would be such that its output $f_\omega(x)$ matches the probability that the user would rate the drum loop x positively. That is,

$$f_\omega(x) \approx P\left(\text{user rating} = \textit{like} \mid \text{input} = x\right). \tag{1}$$

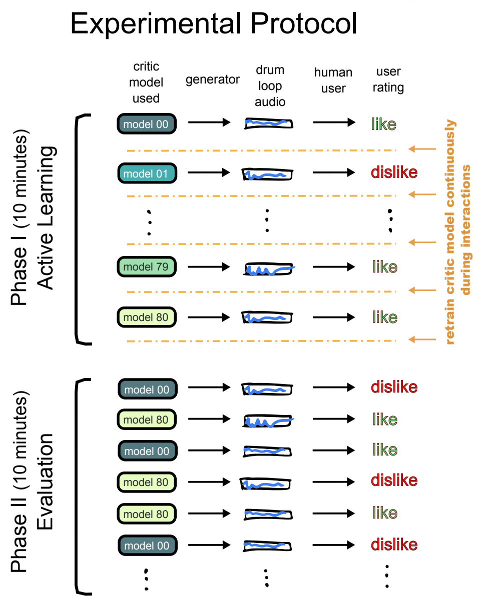

Fig. 2: The experiment is split into Phase I and II.

4 Experimental Protocol

4.1 Hypothesis

We want to demonstrate that, after a minimal number of interactions, DeepDrummer can produce loops that the user is more likely to *like*. The general assumption is that as more interactions are gathered, the critic model will converge towards an accurate approximation of the user's preferences (naturally, this is limited to their preferences within the realm of the drum loops which can be represented by 16 steps with 4 drum one-shot samples).

Because of practical constraints in experiment design, such as limited human attention span, we have intentionally kept the experiment short and minimalistic, and aim to demonstrate that this is something that can be accomplished in merely 10 minutes. With that constraint in mind, we provide a proof of concept of the use of such a method as the core principle behind a more production-ready tool which would naturally involve many more components.

Our goal, thus, is to measure a significant improvement in the proportion of drum loops being *liked* by the users over the course of 80 ratings.

4.2 Phases I & II of the Experiment

The design of our experimental protocol needs to anticipate a potential shift in user preferences (Kulesza, Amershi, Caruana, Fisher, & Charles, 2014; Cartwright

& Pardo, 2016). The issue addressed here does not refer to the randomness of the experiment, for which we can compensate by having many users and ratings. Rather, it points to the fact that a user's behaviour could be influenced by having spent the last 10 minutes rating drum loops. As such, we have designed an experiment with an active learning phase and an evaluation phase, as illustrated in Figure 2.

In Phase I,

- the critic is learning incrementally with every rating from the current user;
- the ratings are used for training the critic and not counted in the analysis afterwards;
- we generate 80 loops, with a probability $p(x) \propto f_\omega(x)$ as in equation (1), meaning that better loops are more common than worse loops. We using the most recent critic model reflecting the latest ratings.

In Phase II,

- the parameters of the critic are fixed;
- the ratings are used purely for evaluation and analysis;
- we generate 60 loops, half from the initial critic and half from the final critic;
- the loops are presented to the user in a randomized order;
- we sample drum loops proportionally to $f_\omega(x)$ again, but we only consider those x such that $f_\omega(x) \geq 0.95$ as a way to present the user with the very best drum loops that we can generate.

Each drum loop generated lasts 2 seconds (one bar at 120 beats per minute) and is repeated 4 times. We also leave one more second of audio at the end for transients to fade. This is done to allow the users to evaluate the rhythmic property of each drum loop. They can, however, give their ratings before the end of the whole sequence. In Phase II, we generate new drum loops that come from the critic models evaluated. We avoid recycling the same drum loops from Phase I as this could lead to specific loops being recognized by the subject. The users are given no information as to which model was used to generate the loops.

5 Results

A total of 25 people participated in this study. We want to determine whether the critic had a better performance at the end than at the beginning, and possibly by how much. Every user i provides 30 binary ratings for their initial critic model and for their final critic model. We are going to analyze the average of those ratings, such that

$$\begin{aligned}
\theta^{(i)}_{\text{init}} &= \text{ratio of drum loops that user } i \textit{ liked} \text{ during Phase II,} \\
& \text{from those generated using the } \textbf{initial} \text{ critic model} \\
\theta^{(i)}_{\text{final}} &= \text{ratio of drum loops that user } i \textit{ liked} \text{ during Phase II,} \\
& \text{from those generated using the } \textbf{final} \text{ critic model} \\
\Delta\theta^{(i)} &= \theta^{(i)}_{\text{final}} - \theta^{(i)}_{\text{init}}.
\end{aligned}$$

By studying the distributions of those two values $(\theta^{(i)}_{\text{init}}, \theta^{(i)}_{\text{final}})$, as well as that of their difference $\Delta\theta^{(i)}$, we can get a sense of the progress made. Larger values of θ correspond to more drum loops being *liked*. In Figure 3a we show the distribution of both $\theta^{(i)}_{\text{init}}$ and $\theta^{(i)}_{\text{final}}$ amongst all the users i. We see that the distribution of θ_{final} has more of its mass around larger values than θ_{init}. A *Wilcoxon signed-rank test* confirms that the values of $\theta^{(i)}_{\text{final}}$ are statistically larger than that of $\theta^{(i)}_{\text{init}}$ (p-value $= 0.00013$). This demonstrates our hypothesis.

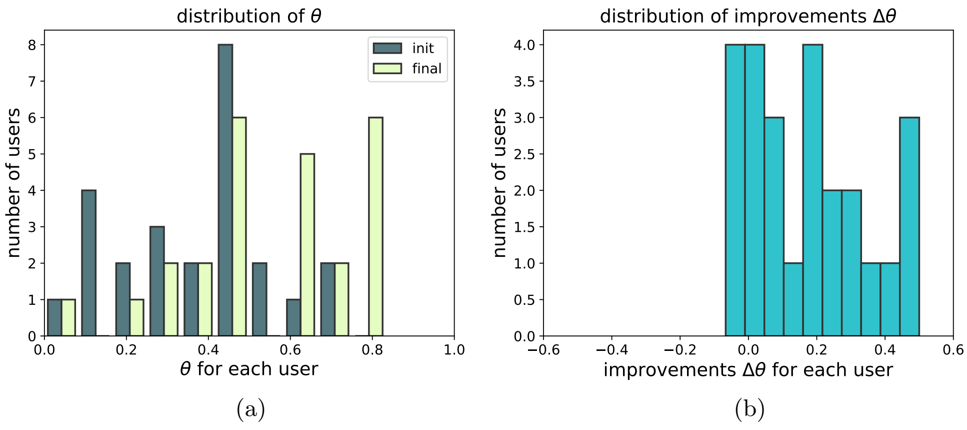

Fig. 3: On the left (a) we compare the proportion of audio loops that users *like* for the initial critic model versus for the final critic model. On the right (b) we report here how much improvement $\Delta\theta^{(i)}$ was achieved. For the majority of users there were clear improvements over the course of a limited number of interactions. A certain percentage of users clustered around 0.0, meaning that they saw no improvement. By combing through our data, we counted 72% of users that have a $\Delta\theta^{(i)} > 0$, and there was 36% of users with $\Delta\theta^{(i)} \geq 0.2$.

6 Conclusion

In this work, we have introduced a novel way of generating music using deep learning with a human in the loop. We presented an implementation of that idea that we call DeepDrummer, which we have shown to very quickly achieve meaningful improvements by proposing drum loops to a user and receiving feedback in the form of *like* / *dislike* ratings.

The simplicity of the idea revolves around the use of a neural network critic that serves as a proxy for the user in order to explore a vast landscape of music very rapidly. Only the most relevant candidates are forwarded to the user.

We ran an experiment with 25 participants in which each user rated 80 drum loops. We have measured empirically the improvement of ratings after those interactions, thus confirming our claims.

Acknowledgments The authors would like to thank Devine Lu Linvega for many enlightening conversations, whether about the world of tools for writing music or the universe in general. We thank Anna Huang for her encouragements and guidance in framing our work as a plausible research project for publication. This research was enabled in part by support provided by Calcul Québéc and Compute Canada.

References

Bogdanov, D., Haro, M., Fuhrmann, F., Xambó, A., Gómez, E., & Herrera, P. (2011). A content-based system for music recommendation and visualization of user preferences working on semantic notions. In *2011 9th international workshop on content-based multimedia indexing (cbmi)* (pp. 249–252).

Cartwright, M., & Pardo, B. (2016). The moving target in creative interactive machine learning. In *Proceedings of the 2016 chi conference extended abstracts on human factors in computing systems (chi ea'16). san jose, california, usa.*

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. In *Advances in neural information processing systems* (pp. 4299–4307).

Davis, S., & Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *28*(4), 357-366.

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., & Sutskever, I. (n.d.). Jukebox: A generative model for music.

Donahue, C., McAuley, J., & Puckette, M. (2018). Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*.

Dong, H.-W., Hsiao, W.-Y., Yang, L.-C., & Yang, Y.-H. (2018). Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Thirty-second aaai conference on artificial intelligence.*

Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., & Roberts, A. (2019). Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*.

Eric, B., Freitas, N. D., & Ghosh, A. (2008). Active preference learning with discrete choice data. In J. C. Platt, D. Koller, Y. Singer, & S. T. Roweis (Eds.), *Advances in neural information processing systems 20* (pp. 409–416). Curran Associates, Inc. Retrieved from `http://papers.nips.cc/paper/3219-active-preference-learning-with-discrete-choice-data.pdf`

Gillick, J., Roberts, A., Engel, J., Eck, D., & Bamman, D. (2019). Learning to groove with inverse sequence transformations. *arXiv preprint arXiv:1905.06118*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

Huang, C.-Z. A., Duvenaud, D., Arnold, K. C., Partridge, B., Oberholtzer, J. W., & Gajos, K. Z. (2014). Active learning of intuitive control knobs for synthesizers using gaussian processes. In *Proceedings of the 19th international conference on intelligent user interfaces* (p. 115–124). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/2557500.2557544` doi: 10.1145/2557500.2557544

Jaques, N., Gu, S., Turner, R. E., & Eck, D. (2017). Tuning recurrent neural networks with reinforcement learning.

Kulesza, T., Amershi, S., Caruana, R., Fisher, D., & Charles, D. (2014). Structured labeling for facilitating concept evolution in machine learning. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 3075–3084).

Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., … Courville, A. C. (2019). Melgan: Generative adversarial networks for conditional waveform synthesis. In *Advances in neural information processing systems* (pp. 14881–14892).

Lattner, S., & Grachten, M. (2019). High-level control of drum track generation using learned patterns of rhythmic interaction. In *2019 IEEE workshop on applications of signal processing to audio and acoustics, WASPAA 2019, new paltz, ny, usa, october 20-23, 2019* (pp. 35–39). IEEE. Retrieved from `https://doi.org/10.1109/WASPAA.2019.8937261` doi: 10.1109/WASPAA.2019.8937261

Lops, P., De Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In *Recommender systems handbook* (pp. 73–105). Springer.

Manzelli, R., Thakkar, V., Siahkamari, A., & Kulis, B. (2018). Conditioning deep generative raw audio models for structured automatic music. *arXiv preprint arXiv:1806.09905*.

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*.

Vogl, R. (2018). *Deep learning methods for drum transcription and drum pattern generation* (Unpublished doctoral dissertation). Ph. D. Dissertation. Johannes Kepler University, Linz, Austria.

Vogl, R., & Knees, P. (2016). An intelligent musical rhythm variation interface. In J. Nichols, J. Mahmud, J. O'Donovan, C. Conati, & M. Zancanaro (Eds.), *Companion publication of the 21st international conference on intelligent user interfaces, IUI 2016, sonoma, ca, usa, march 7-10, 2016* (pp. 88–91). ACM. Retrieved from `https://doi.org/10.1145/2876456.2879471` doi: 10.1145/2876456.2879471

Wang, X., & Wang, Y. (2014). Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd acm in-*

ternational conference on multimedia (pp. 627–636).

Wilson, A., Fern, A., & Tadepalli, P. (2012). A bayesian approach for policy learning from trajectory preference queries. In *Advances in neural information processing systems* (pp. 1133–1141).

Zangerle, E., & Pichl, M. (2018). Content-based user models: Modeling the many faces of musical preference. In *19th international society for music information retrieval conference.*