An Intelligent Drum Machine for Electronic Dance Music Production and Performance

Richard Vogl,[1,2] Peter Knees[1]
[1] Institute of Software Technology & Interactive Systems, Vienna University of Technology, Austria
[2] Department of Computational Perception, Johannes Kepler University, Linz, Austria
richard.vogl@{tuwien.ac.at, jku.at}, peter.knees@tuwien.ac.at

ABSTRACT

An important part of electronic dance music (EDM) is the so-called beat. It is defined by the drum track of the piece and is a style defining element. While producing EDM, creating the drum track tends to be delicate, yet labor intensive work. In this work we present a touch-interface-based prototype with the goal to simplify this task. The prototype aims at supporting musicians to create rhythmic patterns in the context of EDM production and live performances. Starting with a seed pattern which is provided by the user, a list of variations with varying degree of deviation from the seed pattern is generated. The interface provides simple ways to enter, edit, visualize and browse through the patterns. Variations are generated by means of an artificial neural network which is trained on a database of drum rhythm patterns extracted from a commercial drum loop library. To evaluate the user interface as well as the quality of the generated patterns a user study with experts in EDM production was conducted. It was found that participants responded positively to the user interface and the quality of the generated patterns. Furthermore, the experts consider the prototype helpful for both studio production situations and live performances.

Author Keywords

Rhythm pattern generation; restricted Boltzmann machines; machine learning; neural networks; generative stochastic models.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Graphical user interfaces, Input devices and strategies

1. INTRODUCTION

Electronic dance music (EDM) covers a wide range of genres with common production techniques, heavily utilizing synthesizers, sampling, digital effects, and sequencer software or digital audio workstations (DAWs). While these tools are nowadays also used in rock, pop, and other music production processes, they are more prominently featured in and are the foundation of EDM.

An important stylistic property of most EDM genres is the utilization of style-specific repetitive rhythmic patterns, the so-called beat. For shaping and defining the beat, the drum track of the piece and its rhythmic interaction with other instruments are essential. Creating the drum track of an EDM arrangement is, therefore, of high importance and can be time consuming. In this paper we present an intelligent software prototype — implemented as touch interface on a tablet computer — aiming at helping musicians to accomplish this task. More precisely, the developed prototype supports the musician or producer of an EDM track in adding variation to a drum track by intelligently providing creative input for drum pattern creation.

In the prototype's current implementation, we use a step sequencer representation for drum patterns with four drum instruments (kick, snare, hi-hat, open hi-hat) and 16 steps at which these instruments can be either on or off. As artificial intelligence engine, a generative stochastic neural network, concretely a restricted Boltzmann machine trained on EDM drum patterns, is implemented. It is used to generate stylistically suited variations of a given drum pattern. In this context, using loops from drum libraries is usually undesired because it makes the results predictable, boring, and available to everyone (regardless of skill), putting artistic identity in jeopardy. Ultimately, the prototype aims at supporting musicians to create original and interesting rhythmic patterns in the context of EDM production and live performances. To assess the suitability of the presented approach in these tasks, we performed a qualitative user study with expert users in electronic music production. We discuss the outcomes of this study after reviewing related work and detailing the algorithmic and technical implementation.

2. RELATED WORK

There are only a few commercial products for automated rhythmic pattern variation and creation. With Groove Agent,[1] Steinberg provides a drum plugin covering a wide variety of drum kits and loops. It also features a mode which allows variation of the complexity of patterns on the fly. Apple's DAW, Logic Pro,[2] features an automatic drummer plugin which allows the user to select a certain drum kit sound and music style. The patterns played can be changed by controlling complexity and loudness in a two-dimensional system using an x/y pad. These tools aim at amateur and semi-professional production and recording of alternative and rock tracks, and find therefore little use in EDM productions.

In the work of Kaliakatsos-Papakostas et al. [8], a method for automatic drum rhythm generation based on genetic algorithms is introduced. The method creates variations of a rhythm pattern and allows the user to change parameters

[1] http://www.steinberg.net/en/products/vst/groove_agent
[2] http://www.apple.com/logic-pro

NIME'17, May 15-19, 2017, Aalborg University Copenhagen, Denmark.

such as level of variation between the original and the generated patterns. In the work of Ó Nuanáin et al. [9] a similar method for rhythm pattern variation is presented using genetic algorithms in combination with different, more simple, fitness functions. This approach was one of the methods evaluated in [19] where it has been shown that it is more difficult to generate patterns with consistent style using genetic algorithms.

When working with lists of rhythm patterns and sorting or ranking is required, similarity measures for such patterns are needed. In the work of Toussaint [15] several similarity measures for rhythmic patterns are discussed: The Hamming distance, edit distance, Euclidean distance of inter-onset-interval vectors, and the interval-ratio-distance are compared by building phylogenetic trees based on the computed distance matrices. Holzapfel and Stylianou [6] use audio signals as input and present a tempo invariant rhythmic similarity measure utilizing the scale transform. Other methods which can be applied to audio signals are presented by Jensen et al. [7] as well as Gruhne and Dittmar [4]. Both works obtain tempo invariant rhythmic features by applying logarithmic autocorrelation to different onset density functions. Since this work focuses on the use of symbolic representations of rhythm patterns, similarity measures based on audio signals are unsuitable. Therefore, primarily the methods compared in [15] were relevant.

A group of widely used generative models are restricted Boltzmann machines (RBMs) [11, 5]. Battenberg et al. [1] use a variation, the conditional RBM, to analyze drum patterns and classify their meter. They mention the capability of the learned model to generate drum patterns similar to the training data given a seed pattern. Boulanger-Lewandowski et al. [2] use an extension of an RBM with recurrent connections to model and generate polyphonic music. In the work of Vogl and Knees [18] a drum pattern variation method based on an RBM is demonstrated. In [19] different pattern variation methods for drum rhythm generation are evaluated using two user studies. It shows that RBMs are capable of reasonably generating drum patterns.

In the current work, a system which is able to create meaningful variations of a seed pattern is presented. Weaknesses identified in the interface in [19] are considered and a touch-interface-based prototype is introduced. The RBM-based variation method is further improved and the system is evaluated using a qualitative user study involving ten experts in electronic music production.

3. INTELLIGENT PATTERN VARIATION METHOD

The centerpiece of the prototype is the artificial intelligence driven pattern variation engine. Its task is to create rhythm patterns as variations of a seed pattern utilizing sampling of an RBM. For the training of the RBM a data set of EDM and urban music drum rhythm patterns had to be created. RBMs are two layered neural networks which can be used to generate patters. Fig. 1 shows the basic structure and components of a simple RBM. This pattern generation method was chosen for several reasons. Training and sampling of RBMs is well researched and RBMs have been shown to be applicable for pattern generation in general. Furthermore, sampling of RBMs is computationally efficient and can be performed sufficiently fast also on low-end portable devices to ensure reasonable response times for user interaction. When sampling from RBMs, a seed pattern, which determines the characteristics of the generated patterns, can be provided.

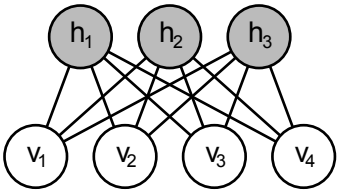

Figure 1: Simplified visualization of the structure of RBMs. The lower layer in white represents the visible nodes (v_n) while the upper layer in gray represents the hidden nodes (h_m). Every connection between two nodes has an assigned weight (w_{mn}) and every node has its own bias value (bv_n and bh_m for visible and hidden nodes respectively – not shown in the diagram).

3.1 Training Data Set

For training, a data set of 2,752 unique one-bar drum patterns containing bass drum, snare drum, and hi-hat onsets was used. The patterns were taken from the sample drum loop library of Native Instrument's *Maschine*[3]. The exported patterns were split into one bar segments, quantized to 16^{th} notes, and converted into a 64 bit (for the 4 by 16 rhythm patterns) binary vector format. Finally, duplicate patterns, as well as patterns which did not meet musical constraints were removed. Only patterns with two to six bass drum notes per bar, one to five snare drum notes, and at least two hi-hat notes were kept. This was done to exclude very sparse breaks as well as too dense fills from the data set. The *Maschine* library contains drum patterns for EDM and other urban music like Hip Hop and RnB. Since the main focus of this work is EDM, this library was well suited.

3.2 Network Training

The used RBM consists of 64 visible nodes, which represent the 16 by 4 drum patterns (16 steps per bar for four instruments), and 500 nodes in the hidden layer. The training for the RBM was performed using the *lrn2* framework of the *lrn2cre8* project[4]. As training algorithm, persistent contrastive divergence (PCD) introduced by Tieleman et al. [14] was used. This method represents an improved version of the well-known contrastive divergence (CD) method introduced by Hinton et al. [5] in 2006. Additionally, latent selectivity and sparsity as described in the work of Goh et al. [3] as well as Drop-out [13] was used to reduce overfitting.

The output of a training run are the weights and biases of the neural network. These are used by the variation algorithm to create the rhythm patterns in the context of the provided seed pattern.

3.3 Pattern Generation

While training of the RBM is similar to [19], the pattern generation was adapted to overcome shortcomings of the method identified in the evaluation of [19].

To use the trained RBM for pattern generation, the seed pattern is first converted to the 64 bit vector format by concatenating the 16 steps of the sequencer grid of each instrument (cf. fig. 2). This vector is then used as input for the visible layer of the RBM. In contrast to [19] no clamping is used and variations are generated for all instruments at

[3] http://www.native-instruments.com/en/products/maschine/production-systems/maschine-studio/
[4] http://lrn2cre8.eu/

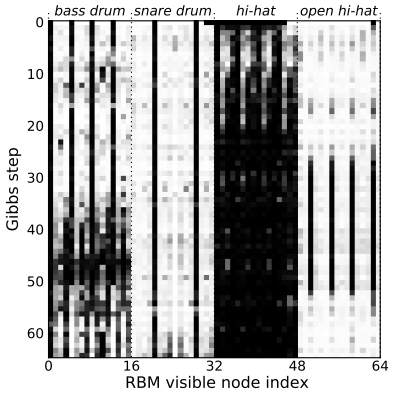

Figure 2: Values of visible layer nodes of the RBM while creating pattern variations. The x-axis represents the index of the visible node. On the y-axis, the number of Gibbs step is indicated starting at the top with the original input pattern and progressing downwards. Black pixels represent 1 and white 0. Only the first 64 iterations of Gibbs sampling are shown.

once using Gibbs sampling.

A Gibbs sampling step consists of: *i.* Calculating the values of the hidden layer (h_m) by multiplying the input layer's values (i_n) with the corresponding weights (w_{nm}) plus bias of the hidden layer (bh_m):

$$h_m = bh_m + \sum_{n=0}^{N} i_n \cdot w_{nm} \tag{1}$$

where N is the total number of nodes in the visible layer. *ii.* Applying the logistic sigmoid function to map the values of the hidden layer into the interval $[0, 1]$ (hs_m) and subsequent binarization with random threshold (sampling):

$$hs_m = \frac{1 + tanh(\frac{h_m}{2})}{2} \tag{2}$$

$$hb_m = \begin{cases} 1, & \text{for } hs_m \geq t \\ 0, & \text{else} \end{cases} \tag{3}$$

where t is a random threshold in the interval $[0, 1]$ and hb_m is the binarized value of h_m. *iii.* Calculating the updated values for the visible layer by multiplying the values of the hidden layer with the corresponding weights (w_{nm}) plus bias of the visible layer (bv_n):

$$i_n = bv_n + \sum_{m=0}^{M} hb_m \cdot w_{nm} \tag{4}$$

where M is the total number of nodes in the hidden layer. To visualize this, fig. 2 shows the values of the nodes in the visible layer during several steps of Gibbs sampling.

In contrast to the pattern variation method used in [19], in this work for every seed pattern 64 variations are generated and only the most suitable (i.e., similar) 32 patterns are presented to the user. This is done to achieve a greater possibility of obtaining an equal number of more dense and more sparse variations. Furthermore, patterns which are very dissimilar to the seed pattern can be discarded.

To arrange the generated patterns in a meaningful way, the patterns are sorted in two steps. First, the patterns are divided into two lists according to the number of active notes in them. One list contains patterns with fewer active notes than the seed pattern (*sparse list*), while the other one contains only pattern with more or equal number of active notes (*dense list*). Second, these lists are sorted according to their similarity to the seed pattern. To build the final list used for the variation dial, the 16 most similar patterns from the *sparse list* are arranged ascending, followed by the seed pattern, followed by the 16 most similar patterns from the *dense list* arranged descending. It should be noted that in rare cases, given a very sparse or dense seed pattern, it may occur that the 64 generated variations do not contain 16 more dense or more sparse patterns. In that case more patterns from the other sub list are used to obtain a final list size of 32 patterns.

3.4 Distance Measure

To sort the pattern lists, a similarity measure for rhythm patterns is required. Although Toussaint [15] observes that the Hamming distance is only moderately suited as a distance measure for rhythmic patterns, it is widely used in the literature (see [9, 10, 19]). Since the requirements in this work are similar to the ones in [19], likewise a modified Hamming distance is implemented. To calculate the standard Hamming distance, simply the differences between the binary vectors of the two rhythm patterns are counted. I.e. for every note in the 16-by-4 grid a check is performed if its state is the same (on/off) in both patterns. If it is not, the distance is increased by one. The modified Hamming distance used in this work weights the individual instruments differently: Differences in the bass drum track contribute with four to the distance, snare drum notes with eight, closed hi-hat notes with one, and open hi-hat notes with four. This is done to take the importance of the drum instruments regarding the perceived differences between rhythm patterns into account. While additional or missing closed hi-hat notes only change the overall rhythmic feel of a pattern very little, additional snare drum or bass drum notes often change the style of the pattern completely. The values for the weighting were determined experimentally and using the results of the web survey in [19].

4. USER INTERFACE

In fig. 3, a screenshot of the prototype's UI is shown. For input and visualization of drum patterns in the UI, the well established *step sequencer* concept is employed. A drum step sequencer, as for example the famous hardware sequencer Roland TR-808, allows the user to activate certain notes for different drum instruments by pressing push buttons in a fixed grid. These patterns are then played back in an endless loop. This concept was used since it is one of the prevalent drum machine interfaces for EDM production. Drum patterns used in this work consist of one bar with notes for bass drum, snare drum, open and closed hi-hat. The time grid resolution is quantized to 16^{th} notes, which is simplification commonly used in step sequencers. The controls for pattern variation are implemented as two buttons (set/reset) and a central dial on which the variations are placed ordered by sparsity and distance to the seed pattern. Variations are generated by pressing the set button. The seed pattern is expected not to be highly complex, nor too simple, to give the variation algorithm enough freedom to find variations in both directions. During exploration of the patterns, the seed pattern can always be quickly accessed by pressing the "reset" button.

Next to the pattern variation controls a start/pause playback button and knobs to control the tempo in beats per minute (BPM) and the ratio of swing for 8^{th} notes can be found. The selected tempo and swing-ratio do not affect the pattern variation but rather provide the possibility to

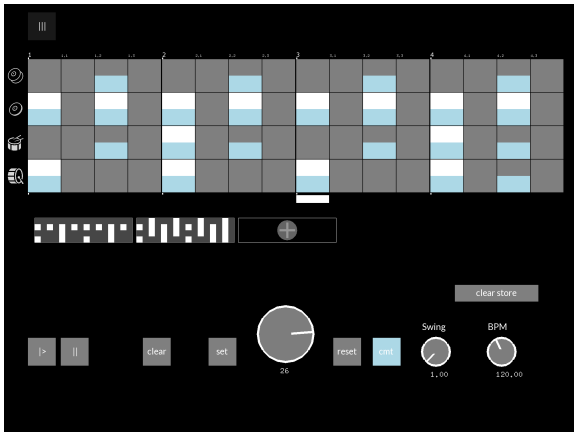

Figure 3: Screenshot of the UI. It consists of a 4 by 16 button array which poses as the visualization and input of the step sequencer. Beneath it are pattern storage, controls for playback, the pattern variation controls, and controls for tempo and swing. The blue blocks in the lower half of a step sequencer block visualize the pattern under preview. It can be activated by pressing the blue commit button in the pattern variation control section of the UI.

tune the rendering of the entered beat to match the musical style desired by the user.

The utilization of a touch-base interface allows the user to enter and change patterns in the step sequencer grid more easily and without the use of a mouse or track pad. This is expected to improve the acceptance of such a tool in live environments where simplicity and robustness of the input methods are obligatory. On the other hand this means that a widespread and accepted input method, namely physical knobs of MIDI controllers are not necessary, since all knobs can be controlled via the touch surface. While it is possible to map the input knobs and buttons to external MIDI controllers, in this work, another input method, namely knobs which can be used on the touch surface were evaluated. Fig. 4 shows the usage of such knobs on an iPad running the software prototype. The concept of physical knobs which can be used on a touch interface is not new (see e.g. ROTOR[5] or Tuna Knobs[6]). Using such knobs allows the combination of two powerful input methods with individual strengths. While the multi-touch interface provides an easy way of manipulating rhythm patterns, physical knobs might provide a greater degree of precision than fingers on a touch interface.

Improvements proposed in [19] cover: *i.* a pattern storage, *ii.* an optional pattern preview system, and *iii.* the possibility to activate new patterns only at the start of a new bar.

The pattern storage which is located beneath the step sequencer grid can be used to store patterns by tapping the plus sign. Stored patterns are displayed as a list of small thumbnails of the sequencer grid and can be brought back to the step sequencer by tapping the patterns. The store can be cleared using the "clear store" button beneath the store grid.

The pattern preview function is integrated into the step sequencer. When the preview is switched on, the active pattern is depicted using white blocks in the sequencer. If the variation dial is used to browse through the variations,

[5] http://reactable.com/rotor/

[6] http://www.tunadjgear.com/

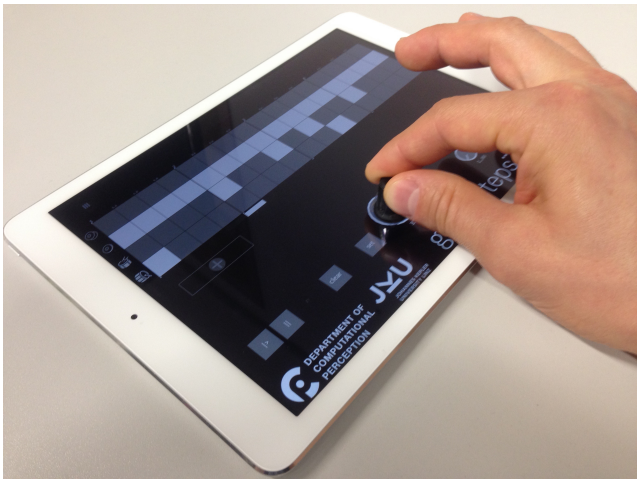

Figure 4: A physical knob used to control the variation dial on the touch surface. While the used knob is an improvised prototype, commercial products already exist for this purpose.

they are not immediately used for playback but rather visualized as blue blocks in the lower half of the step sequencer grid. See fig. 3 which shows a screenshot of a pattern being visualized in preview mode while another pattern is active. To use the currently previewed pattern, the commit button ("cmt") has to be pressed.

The prototype can be synchronized with other instruments using Ableton's Link technology.[7] The output of the prototype is sent via MIDI to be further processed in a DAW or synthesized using an external MIDI instrument. Alternatively, an internal drum synthesizer can be used to render audio playback of the patterns.

5. EVALUATION

To evaluate the interaction with the prototype as well as the quality of the generated patterns compared to the prototype presented in [18], a qualitative user study was conducted. To this end, experts were interviewed using a questionnaire as guideline. The experts were required to have experience in *i.* using DAWs or similar music production software, *ii.* producing or performing electronic music live, and *iii.* using drum step sequencers and/or drum roll editors. The software prototype used in [19] was made available by the authors as accompanying materials of the article[8].

During a session, participants were introduced to the two prototypes and the aim and functionality was explained. They were asked to input rhythm patterns they usually work with and let the prototype generate a list of variations for these patterns. After browsing through the patterns and exploring the features of the systems, users were interviewed about their experience with the prototypes and their preferences. Specifically, they were asked to rate the following properties on five point Likert scales: *i.* The usability of the prototypes, *ii.* the application of such a tool in a live performance or *iii.* in a studio production environment, *iv.* preferred input method (MIDI controller and mouse, touch interface, or touch interface combined with physical knobs), and *v.* usefulness of the additional features in the touch-interface-based prototype.

Additionally to the UI evaluation, the differences between the pattern variation algorithms were also tested. To this

[7] https://www.ableton.com/en/link/

[8] https://github.com/GiantSteps/rhythm-pattern-variation-study

	Vogl et al. [19]	Present work
Consistency	3.7	**3.9**
Musicality	4.2	**4.4**
Difference	3.2	**2.9**
Difference RMSE	0.6	**0.3**
Interestingness	3.8	**4.0**
Substitute	3.8	**4.4**
Fill	**4.0**	3.6

Table 1: Mean values of participant rating for the two algorithms. For *difference* additionally the RMSE to the neutral value (3) is provided.

end, both algorithms were implemented in the touch-interface-based prototype. Participants were asked to browse through variation lists generated by both algorithms for seed patterns of their choice. After that, they were asked to rate both algorithms in the following categories on five point Likert scales: *i.* Consistency of the variations with the seed pattern, *ii.* musicality and meaningfulness of created patterns, *iii.* difference of created patterns to the seed pattern, *iv.* creativity and interestingness of created patterns, *v.* suitability of created patterns for a continuous beat, and *vi.* suitability of patterns for fills or breaks. These categories correspond roughly to the ones used in the web survey in [19]. The order in which the algorithms were tested was randomized to avoid experimenter bias. The Likert scale for the *difference* rating ranged from "too similar" to "too different", therefore the optimal answer "just right" was placed in the middle. This is also reflected in the evaluation section: For the *difference* ratings additionally root mean square errors (RMSE) towards the optimal value (3) are provided.

A more general discussion including topics like the prototype's concept and applicability, positive and negative experiences during the experiment, UI details, missing features, and the participant's usual workflow and preferred tools concluded the sessions.

6. RESULTS AND DISCUSSION

The interviews were conducted during the period between June and October of 2016. In total ten experts participated in the survey. Their mean age is 31.1, the gender distribution is 9 male and one self-identified neither as male nor female. Seven participants had formal musical education whereas three are autodidacts. Eight participants actively play an instrument and all use DAWs on a regular basis, are familiar with step sequencers and have several years of experience in electronic music production.

Tab. 1 shows the mean values for the participants' ratings of the comparison between the variation algorithm used in [19] (top) and the one presented in this work (bottom). Since the number of participants of ten is too low for meaningful statistical significance analysis the numbers merely indicate tendencies. Nevertheless, a Wilcoxon signed ranks test was used to test for significance in the rating differences in aspects of the two UIs, and the two variation algorithms. As expected, the observed improvements are not significant (alpha=.05) due to small sample size, except for assessment of usability, where the touch based UI was considered better usable. The ratings in combination with in depth discussions with the participants show a clear preference towards the UI and variation generation algorithm presented in this work. The only exception being the suitability of the generated patterns for fills. This can be explained by the fact that outlier patterns are discarded by the variation algorithm and therefore it produces patterns more similar to the seed pattern, which may be less suitable for fills. The exact definition of fills depends on the music style, e.g. the "amen-break" is originally a fill but forms the basis of the continuous rhythms in drum-and-bass and breakbeat music. Generally, patterns which greatly differ from the basic rhythms, but somehow fit the given style can be considered as fills and breaks. For the tasks at hand, we relied on the individual understanding and definition of the expert users.

Tendencies regarding the ratings for the UI are similarly consistent. Ratings for usability are significantly higher for the touch interface (mean: 4.7/4.3). The difference is even greater for the suitability in live scenarios (mean: 4.0/3.5). While 50% of the participants uttered concerns about the practicality of using a mouse to enter rhythm patterns on stage, only two participants were concerned that the touch device is not suitable for a live performance. One participant's reservations regarding the touch interface did not concern the way of interaction but rather if the hardware (iPad) would survive the harsh conditions on stage and on the road (heat, mechanical strain, spilled drinks, etc.). The second one raised concerns regarding the touch interface's precision and reliability in a live environment. Regarding the applicability of the prototypes in a studio or production setting, the difference was smaller, but still in favor of the touch based prototype (mean: 4.7/4.6). The comment of one participant nicely summarizes the tenor of the users:

> "Using the touch interface is definitely faster and easier [...] compared to entering patterns with a mouse." Participant03

Regarding the preferences of the input method, a clear tendency towards the touch interface was observable: Six participants preferred the touch interface, three were undecided, and only one voted in favor of the physical controller and mouse system. Regarding the touch-compatible physical knob prototypes, seven participants preferred the touch-only approach, one was undecided, and two preferred using the physical knobs.

The additional features were generally received very positively. Only two participants were unsure if the feature to start new patterns only with a new bar was useful. All other participants were in favor for all three additional features.

In the discussions with the participants several key messages were identified. Three participants considered the arrangement of the patterns in the one-dimensional list of the variation wheel as being unclear or ambiguous:

> "It seems a bit random to me. I can browse through the list [...] but I cannot look for something specific." Participant04

While the idea of a simple one-dimensional variation dial introduced in [18] was well suited to conduct experiments regarding the quality of variation algorithms, it might be an over-simplification for user interaction. After all two different properties (sparseness and similarity) are projected into one dimension. Participants suggested to solve this by adding the option to change the sorting of the list or by using an x/y variation pad similar to the one used for the Drummer plugin of the Logic Pro DAW.

While the visual preview was a well received feature, two participants missed an acoustic preview or "pre-listen" mode. Finding suitable audio material for remixing by listening to it on separate headphones is a common technique used by DJs in live situations.

Almost all participants (8/10) mentioned that they use a drum roll editor within their DAW to produce drum rhythm patterns. One explicitly stated that he tries to avoid it:

> "I use the piano roll editor in Cubase if I have to, but it is a real pain." Participant04

7. CONCLUSION

In this work we presented a touch-interface-based prototype to assist musicians and producers in the context of EDM production with creating the drum track. This is accomplished by providing variations of a basic rhythmic pattern entered by the user. The variations are created utilizing Gibbs sampling of an RBM trained on appropriate rhythm patterns. The implemented method builds on established algorithms and improves them. Accessing and browsing through the patterns is accomplished using a simple interface built around a central dial.

A user study was conducted to evaluated both the pattern variation algorithm as well as the user interface of the prototype. The results of the study show that musicians and producers consider the interface intuitive. It is also shown that acceptance of the system in a live scenario is higher than for the compared prototype while acceptance in a production setting is still given. The pattern variation algorithm was considered to produce patterns more consistent with the seed pattern than the compared system, but only at the expense of the capability to create patterns suitable for fills and breaks.

The UI incorporates additional features requested by participants of a similar study. These features cover a preview for generated patterns, a pattern storage, and the ability to start patterns only at bar changes. These features were received positively by the participants. While the idea of using physical knobs on a touch interface was interesting for many participants, most participants preferred using a simple touch interface without additional knobs, especially in live settings.

Considering the feedback of the expert users, it can be concluded that such a system could find acceptance in the area of EDM production and performance. To achieve this, the prototype will still have to be improved in regard of the UI's visual design as well as the arrangement and browsing metaphor of the drum patterns.

To support more drum instruments prevalent in EDM production a larger and more diverse training data set for the variation method will be necessary. To obtain such a data set, automatic drum transcription methods (e.g., [17, 12, 16, 20]) could be utilized. Using such an approach would also allow to expand this method to be applicable on other music genres outside of EDM. However, to gain acceptance in the community of producers and musicians of other music genres an entirely different approach for pattern visualization and input might be required, since the used step sequencer representation is, at the moment, prominently tied to the field of EDM production. Nonetheless, from the results obtained in this study, we are confident that the use of generative models provides a valuable addition to the current paradigms of music production practice and helps in building intelligent and therefore more intuitive and effective interfaces.

8. ACKNOWLEDGMENTS

We thank Matthias Leimeister for extracting the MIDI and text mapping for the *Maschine* sample drum loop library. This work was supported by the European Union FP7 through the GiantSteps project (grant agreement no. 610591) and the Austrian FFG under the BRIDGE 1 project SmarterJam (858514).

9. REFERENCES

[1] E. Battenberg and D. Wessel. Analyzing drum patterns using conditional deep belief networks. In *Proc 13th ISMIR*, 2012.

[2] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proc 29th ICML*, 2012.

[3] H. Goh, N. Thome, and M. Cord. Biasing restricted boltzmann machines to manipulate latent selectivity and sparsity. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[4] M. Gruhne and C. Dittmar. Improving rhythmic pattern features based on logarithmic preprocessing. In *Proc 126th AES Convention*, 2009.

[5] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, July 2006.

[6] A. Holzapfel and Y. Stylianou. Scale transform in rhythmic similarity of music. *IEEE TASLP*, 19(1):176–185, 2011.

[7] J. H. Jensen, M. G. Christensen, and S. H. Jensen. A tempo-insensitive representation of rhythmic patterns. In *Proc 17th EUSIPCO*, 2009.

[8] M. A. Kaliakatsos-Papakostas, A. Floros, and M. N. Vrahatis. evodrummer: Deriving rhythmic patterns through interactive genetic algorithms. In *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, *LNCS* vol 7834, 2013.

[9] C. Ó Nuanáin, P. Herrera, and S. Jordà. Target-based rhythmic pattern generation and variation with genetic algorithms. In *Proc 12th Sound and Music Computing Conference*, 2015.

[10] J.-F. Paiement, Y. Grandvalet, S. Bengio, and D. Eck. A generative model for rhythms. In *NIPS Workshop on Brain, Music and Cognition*, 2007.

[11] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 194–281. MIT Press, 1986.

[12] C. Southall, R. Stables, and J. Hockman. Automatic drum transcription using bidirectional recurrent neural networks. In *Proc 17th ISMIR*, 2016.

[13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, June 2014.

[14] T. Tieleman and G. Hinton. Using fast weights to improve persistent contrastive divergence. In *Proc 26th ICML*, 2009.

[15] G. Toussaint. A comparison of rhythmic similarity measures. In *Proc 5th ISMIR*, 2004.

[16] R. Vogl, M. Dorfer, and P. Knees. Recurrent neural networks for drum transcription. In *Proc 17th ISMIR*, 2016.

[17] R. Vogl, M. Dorfer, and P. Knees. Drum transcription from polyphonic music with recurrent neural networks. In *Proc 42nd ICASSP*, 2017.

[18] R. Vogl and P. Knees. An intelligent musical rhythm variation interface. In *IUI Companion*, 2016.

[19] R. Vogl, M. Leimeister, C. Ó. Nuanáin, S. Jordà, M. Hlatky, and P. Knees. An intelligent interface for drum pattern variation and comparative evaluation of algorithms. *JAES*, 64(7/8):503–513, 2016.

[20] C.-W. Wu and A. Lerch. Drum transcription using partially Fixed Non-Negative Matrix Factorization with Template Adaptation. In *Proc 16th ISMIR*, 2015.