Neural Computing and Applications
https://doi.org/10.1007/s00521-018-3708-6

S.I. : EANN 2017

Conditional neural sequence learners for generating drums' rhythms

Dimos Makris[1] · Maximos Kaliakatsos-Papakostas[2] · Ioannis Karydis[1] · Katia Lida Kermanidis[1]

Received: 8 January 2018 / Accepted: 10 September 2018

Abstract
Machine learning has shown a successful component of methods for automatic music composition. Considering music as a sequence of events with multiple complex dependencies on various levels of a composition, the long short-term memory-based (LSTM) architectures have been proven to be very efficient in learning and reproducing musical styles. The "rampant force" of these architectures, however, makes them hardly useful for tasks that incorporate human input or generally constraints. Such an example is the generation of drums' rhythms under a given metric structure (potentially combining different time signatures), with a given instrumentation (e.g. bass and guitar notes). This paper presents a solution that harnesses the LSTM sequence learner with a feed-forward (FF) part which is called the "Conditional Layer". The LSTM and the FF layers influence (are merged into) a single layer making the final decision about the next drums' event, given previous events (LSTM layer) and current constraints (FF layer). The resulting architecture is called the conditional neural sequence learner (CNSL). Results on drums' rhythm sequences are presented indicating that the CNSL architecture is effective in producing drums' sequences that resemble a learnt style, while at the same time conform to given constraints; impressively, the CNSL is able to compose drums' rhythms in time signatures it has not encountered during training (e.g. 17/16), which resemble the characteristics of the rhythms in the original data.

Keywords LSTM · Neural networks · Deep learning · Rhythm composition · Music information research

1 Introduction

Computational systems that can claim to be creative [8] have long been the focus of research. Despite the inherent difficulty of formally defining the notion of creativity, numerous working definitions exist and equivalently various such systems [4].

Music has received significant attention in relation to other arts such as graphical arts, painting, dance, literature or architecture due to its rigorous formalisation, available from the early stages of its evolution [29] as well as for its advantageous quality to not require the complicating mechanisms of referential meaning in its study [8].

The seminal work of Hiller and Isaacson [14], on the composition of a musical piece using a computer program, was published as early as shortly after the introduction of the very first computer while Wiggins et al. [35] present a detailed account of various musically creative systems.

The spectrum of musical creativity is fascinating and certainly wide, encompassing creativeness in musical theory/philosophy, listening, education, performance and therapy among others [8]. Creativeness in musical composition represents one of the four major areas of musical creativity [26].

Accordingly, the musical composition process can be mapped to an algorithmic approach [17, 22]. Among numerous definitions of algorithmic (or automatic) music

This research has been financially supported by General Secretariat for Research and Technology (GSRT) and the Hellenic Foundation for Research and Innovation (HFRI) (Scholarship Code: 953).

✉ Dimos Makris
c12makr@ionio.gr

Maximos Kaliakatsos-Papakostas
maxk@mus.auth.gr

Ioannis Karydis
karydis@ionio.gr

Katia Lida Kermanidis
kerman@ionio.gr

[1] Department of Informatics, Ionian University, Corfu, Greece

[2] Institute for Language and Speech Processing, R.C."Athena", Athens, Greece

Published online: 20 September 2018

composition (AMC) [17, 34], D. Cope[1] provided an interesting alternative, "a sequence (set) of rules (instructions, operations) for solving (accomplishing) a [particular] problem (task) [in a finite number of steps] of combining musical parts (things, elements) into a whole (composition)" [31].

AMC is a complex problem with tasks ranging from melody and chords' composition to rhythm and lyrics, among others [5]. AMC applications include systems with varying degrees of combination of computational creation (e.g. fully automated background harmonisation music or assisted co-composition) with humans' creativity for the production of musical works [11]. AMC has been approached with a plethora of methods from various points of view of artificial intelligence methods such as grammars, symbolic and knowledge-based systems, Markov chains, artificial neural networks, evolutionary population-based as well as self-similarity and cellular automata [11].

Artificial neural networks (ANNs) acted as a powerful computation tool to extend the available methods. The first attempts were usually focused on the four-part (Chorale) harmonisation task (e.g. [7, 13]) by proposing pure neural architectures or combinations (hybrid approaches) with ruled-based systems. However, the first impressions were not remarkable and probabilistic approaches still had better results by that time [2]. With the advent of deep learning architectures, numerous research works have been published using different types of ANNs and, especially recurrent neural networks (RNNs) for composing music (e.g. [12, 20]).

1.1 Motivation and contribution

Recurrent neural networks, and especially the long short-term memory (LSTM) networks, have been utilised for music generation (see [33] for further references), since these are capable of modelling sequences of events. However, hitherto proposed architectures (e.g. [16]) focused on the generation of sequences per se, without considering important constraints.

For instance, the performance of human drummers is potentially influenced by what the guitar and bass player plays, while the tempo of a song affects the density of their playing. Additionally, human drummers could play drums on a time signature they have never played before, e.g. 15/8, by utilising the knowledge they have obtained only by practicing (learning) 4/4 beats; this happens because human drummers have an a priori understanding of metric information which helps them perform drums' rhythms on metres they have never seen before.

Using the LSTM architecture, it is methodologically impossible to compose drums' sequences in unknown time signatures for the learnt style, or follow other instruments. To address the aforementioned weaknesses of the previously proposed methodologies, this work proposes the combination of different neural network layers, i.e. feed-forward (FF) and LSTMs, for composing drums' sequences based on musical parameters, i.e. guitar and bass performance, tempo and grouping indications, and metric information. The resulting architecture is called conditional neural sequence learners (CNSLs).

The CNSLs are able to combine implicitly learnt information (LSTM module) and explicitly defined conditions (FF module), allowing the generation of musical sequences (drums' sequences in the herein examined case) that resemble a musical styles through implicit learning and, at the same time, satisfy some explicitly declared conditions that are potentially not encountered in the learnt style. This fact gives the proposed architecture a clear advantage: complex sequence-related rules (e.g. rhythm relations between accompanying instruments and drum generation) do not need to be explicitly formulated (since these are implicitly learnt), while at the same time employing explicit information is allowed.

The remainder of this paper is organised as follows: Sect. 2 presents background information and existing research on automated musical composition and related notions. Next, Sect. 3 presents the proposed combination of LSTM and feed-forward neural networks for conditional rhythm composition. Section 4 details the experimental evaluation of the proposed method, while the work is concluded in Sect. 5.

2 Related work

Following the definition of creativity by Jacob [17], in a post-digital computer age, algorithmic musical composition (AMC) mostly refers to a process of incremental and iterative revision similar to hard work in contrast to creativity being the result of inspiration or genius similar to a "flash out of the blue". The use of artificial neural networks (ANNs) has long be proposed for automated composition.[2] The capability of ANNs to resemble human creative activities has been the main reason for their use in AMC despite the extensive training required by ANNs in order to do so.

AMC is fundamentally based on the ground-breaking work of Guido d'Arezzo who invented a formal music notation, approximately a millennium ago [10]. The

[1] Panel Discussion in the ICMC '93.

[2] Papadopoulos and Wiggins [31] collected an extensive such use list, dating back to 1992.

evolution of the notation led to a set of rules and frameworks that guide the creation process of musical composition [28]. Accordingly, music may be represented as a sequence of events that feature modelled probabilities between them [5], leading thus to the prediction of current events based on past events and potentially the use of extraneous information.

A plethora of methodologies have been proposed for the purposes of ACM such as mathematical models, knowledge-based systems, grammars, evolutionary methods, systems which learn, and hybrid systems [31], while many of these areas are indeed intertwined. Despite the wide range of approaches in the domain, a few open problems persist even nowadays: the requirement for incorporation of human domain experts during the evaluation phase, the trade-off between knowledge representation and search methodology as well as the mapping of human creativity, a creativity with vague and case-specific definition, to the ACM methods.

The feed-forward ANN is a very common version of ANN, wherein neurons/units are commonly divided into three types of layers, namely input, hidden, and output. Interconnection of units between different layers is achieved by use of weights that are multiplied with the values of their respective input unit. The resulting output is then propagated to a transfer function through units to the output unit. In feed-forward ANNs, the "learning" procedure describes their ability to modify the aforementioned connecting weights in order to optimally match known target values, for the scenario of supervised learning, by use of error minimisation. Similarly to ANNs, recurrent neural networks (RNNs) share a relatively common architecture though hidden layers exhibit a larger degree of sophistication by featuring the ability to include connections that remember past events. AMC using RNNs has been extensively proposed for purposes such as generation of chord sequences [23] and melodies [27].

However, recurrent networks suffered from training problems (*vanishing* or *exploding* gradient problem) which was addressed by Hochreiter and Schmidhuber [15] and resolved with the long short-term memory (LSTM) architecture which is an initial form of the RNN architecture. As this solution has been effective, the LSTMs prevailed and became the standard for recurrent networks. Following the principles of RNNs, LSTM networks are generic in the sense that given adequate network units, they allow for any conventional computation. In contrast to RNNs, LSTMs are more suited to learning from experience for classification, as well as process and prediction of time-series, when there are time lags of unknown size bounded between important events. LSTMs feature relative insensitivity to gap length, thus being advantageous to alternative forms of RNNs or hidden Markov models.

Accordingly, LSTMs have been utilised extensively for AMC purposes such as learning chord progressions as well us creating musically appealing four-part chorale harmonisations. Eck and Schmidhuber [9] describe experiments for learning chords along with accompanying melodies in the blues style using multiple LSTM blocks. The generation is performed by seed notes and chords while the authors reports satisfying results. The BachBot [24] system is a pure recurrent network architecture with the objective to generate polyphonic music in the style of Bach's Chorales. The proposed architecture consists of three LSTM layers while the generation is done by giving an initial soprano melody seed. Hadjeres et al. [12] proposed DeepBach, a complex deep neural network architecture, specialised again for Bach's Chorales. DeepBach consists of two recurrent (LSTMs) blocks used for summing up past and future information together with a feed-forward layer providing information for the current events. The three outputs are then merged and passed on to a final feed-forward network. Generation is done by sampling, using a pseudo-Gibbs procedure while the results presented are really promising. Finally, more information for different topologies and strategies can be found in the deep learning music survey [3].

To the best of our knowledge, there are limited works attempting learning and generation of percussion (drum) progressions with neural networks. Choi et al. [5] introduced an LSTM-text-based approach for automatic drums' generation. In order to encode simultaneous drum events into texts for training data, they used binary representation of pitches. The authors generated drums' sequences with seed sentences and reported promising results. In [16], Hutchings presents a method for the generation a full drum kit part given a kick-drum sequence using a "sequence-to-sequence" neural network model that encodes multiple musical styles. Results indicated that use of a sampling technique drawing from the three most probable outputs at each subdivision of the drum pattern were promising, but heavily dependent on style. Most importantly though, and similarly to the rest of existing works, Hutchings [16] employs solely raw drums' information in contrast to our proposed method, rendering it thus not comparable to our approach.

3 Conditional rhythm composition with LSTM and feed-forward neural networks

Herein, we introduce a novel architecture for creating drums' sequences by taking into account the drum generation of previous time-steps along with additional musical information regarding metric structure, tempo, grouping

and instruments playing. Section 3.1 describes the collection and preprocessing methodology for the training data, while Sect. 3.2 presents the two proposed model's architectures.

3.1 Data collection and architecture information

The utilised corpus consists of 105 excerpts of musical pieces, featuring three bands with average length 2.5 min each, which were collected manually from web tablature learning sources.[3] During the conversion to MIDI files, preprocessing was applied in order to maintain the drum, bass and single guitar tracks. For each song, selected characteristic snippets were marked as different phrases (e.g. *couple*, *refrain*). The phrasing selections and the annotations were conducted with the help of students of the Music Department of the Ionian University, Greece.

The dataset is separated to three parts with 35 pieces from each band plus five synthetic pieces with phrases including Toms and Cymbals events due to the fact they had very few appearances in the learning data. Band A (denoted as *AB*) follows the "80s Disco" style while Band B (denoted as *PF*) follows "70s Blues and Rock" with Band C (denoted as *PT*) having the most contemporary style featuring "Progressive Rock and Metal".

We use two different input spaces to represent the training data and feed these into two different types of ANNs. The first one, an LSTM network, corresponds to the drums' representation, while the second one, the feed-forward network, represents information of the bass and guitar movement, the metrical structure, tempo and the grouping (phrasing) information. Standard MIDI notation form is used with the time quantisation set to 1/16th note.

3.1.1 LSTM layers' representation

For the LSTM layers' data input representation, we use the *item* or *one-hot* encoding which has been used extensively in similar research [5, 25]. The key idea is to consider each possible pitch note as a distinct element (token) of a vocabulary. Considering N distinct input tokens, then the presence of a note pitch will be encoded as value 1 for the corresponding input token and value 0 for all the rest tokens. As far as the drums are concerned, two different representations are used which lead us to two variations of the proposed architecture. In addition, we increased the vocabulary with two elements acting as flags, indicating the beginning (pitch value -1) and the ending (pitch value 128) of every training piece. Finally, we only take into account pitch and onset time parameters of every MIDI event; thus, offset times (ending positions of notes) are not calculated.

Following the same methodology for conditional composition, similarly to Makris et al. [25], the representation is based on text words, as proposed by [5]. To encode simultaneous events in a track into texts, binary representation of pitches is used, i.e. standard components of drums-kick, snare, hi-hats, cymbals and toms. Due to the expansion of the training data, and in order to achieve efficient representation and learning, we increase from 5 to 11 the retaining components: kick, snare, closed and open hi-hat, ride cymbal, three tom events, two crash and one china cymbal. There can be theoretically $2^{11} = 2048$ words, but since the combinations of drum components in pieces of the dataset that are actually played are limited, these were restrained to 97.

Moreover, herein a variation of the drums' input data representation is introduced leading to multiple LSTM blocks, each one responsible for learning different drums' components. The training data is separated to three categories, with *one-hot* encoding; $Input_1$ for kick and snare, $Input_2$ for hi-hat and ride events and $Input_3$ for tom, crash and China events with corresponding 7, 8 and 13 number of features, respectively.

3.1.2 Feed-forward representation

Moving on to the feed-forward (FF) input space, combinations of *one-hot* encodings of every musical aspect were taken into account regarding guitar, bass, tempo and grouping and metrical structure information with an overall set of 26 features. Table 1 summarises all the proposed FF musical features.

As far as the metric structure is concerned, a 1×3 *binary vector* is included representing metrical information for each time-step. This information ensures that the network is aware of the beat structure at any given instance. Its first digit declares the start of a bar, the second digit the start of half-bar, while the third digit, the start of quarter bar. For example, the first nine time-steps of a sequence with time signature of 3/4 can be translated to metrical information as: [111], [000], [000], [000], [001], [000], [010], [000], [001]. In the training corpus, 14 different

Table 1 Feed-forward network features

Feature description	Feature indexes
Guitar performance and polyphony	1–9
Bass performance	10–13
Tempo information	14–18
Metrical structure	19–21
Grouping information	22–26

[3] http://www.911tabs.com/.

time signatures were detected. The most common was 4/4, though more complex time structures can be found, such as 9/8 or 15/8.

Moving on to the guitar and bass, we used information regarding the polyphony of each instrument and whether it performs or not in each time-step. Since from the perspective of a drummer the information about actual pitch notes is not directly involved in the beat structure, this feature set's purpose was to assist the network in generating drums' events by also considering, during learning, when onsets on the bass and guitar occur (i.e. learning the contribution of each instrument to the overall rhythm with their onsets and rests). Moreover, the polyphony of the guitar track can be useful due to the ability of the network to learn how to perform when encountering chords or single notes.

Tempo information potentially allows the network to adjust the density of the events it produces (e.g. slower tempos might tolerate more events within a bar in comparison to faster tempos). Thus, the network is aware of changes during a song, which may lead to change the style of playing, while the grouping part contains annotations about structurally coherent temporal regions of the music (i.e. different couples or refrains). Moreover, flags for the first and the last measure of every phrase were included. The use of grouping indicators allows the network to behave properly and learn the initial and final drum events within a phrase.

3.2 Proposed architecture

The proposed architecture consists of two separate modules for predicting the next drum event. The LSTM module learns sequences of consecutive drum events, while the feed-forward module handles musical information regarding guitar, bass, metrical structure, tempo and grouping. This information is the sum of features of consecutive time-steps of the feed-forward input space within a moving *window* giving information about the past, current and future time-steps. The length of the window depends of the memory of the LSTMs (*Sequence Length*) and has value of $2 * \texttt{Sequence Length} + 1$ time-steps. Since most of the training examples are in 4/4 time signature, the memory of the LSTMs is predefined with the fixed value of 16 time-steps corresponding to a full bar.

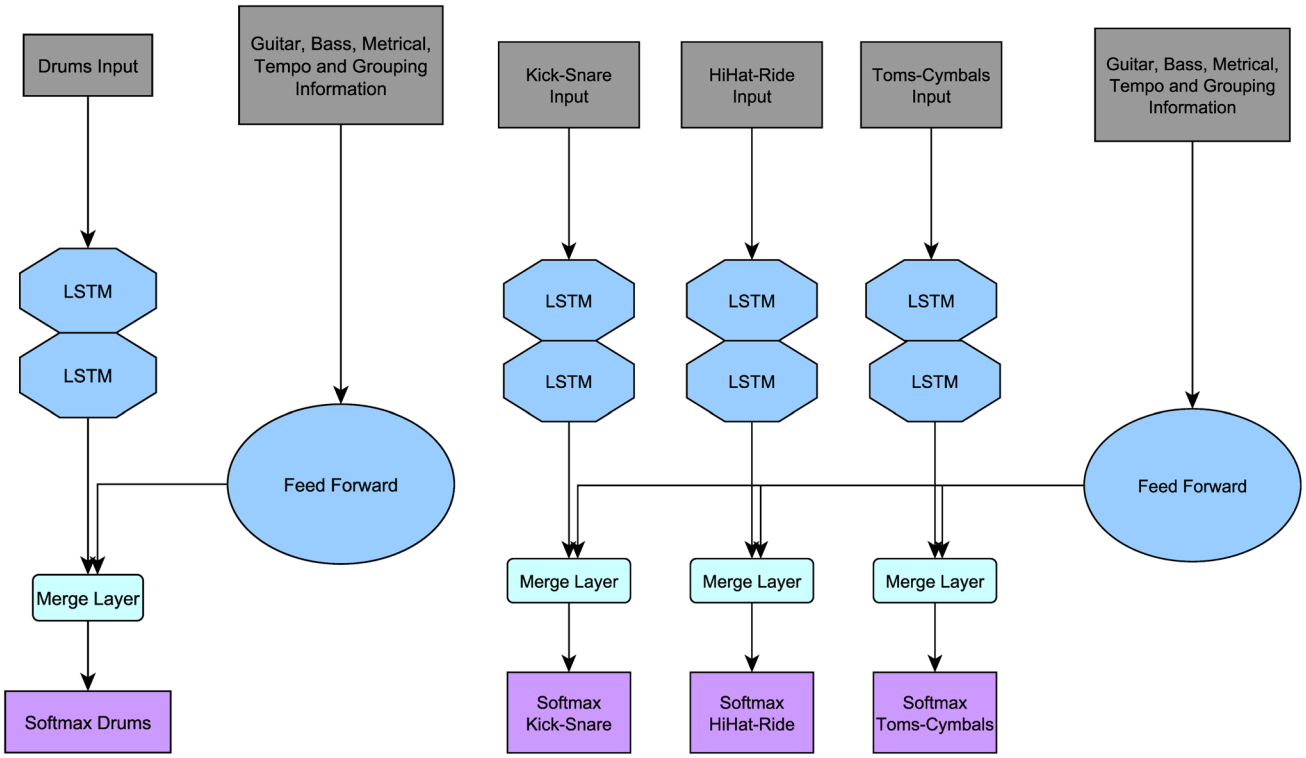
For our experiments, we use two variations of the proposed architecture depending on the different LSTM representation input data as described in Sect. 3.1.1. The first one, denoted as $Arch_1$, is the same as it was used on our prior work [25] and it consists of a single LSTM layer block merged with the feed-forward layer on a single *softmax* output. The second variation, $Arch_2$, consists of four input spaces with three LSTM layer blocks, each one responsible for different drum's *Inputs* and the feed-forward layer combined into different merge layers thus leading to independent softmax outputs. The resulting architecture is called the conditional neural sequence learner (CNSL) and Fig. 1 summarises both variations.

As far as the configuration for LSTM blocks is concerned, we use two stacked LSTM layers with 256 Hidden Units and a dropout of 0.2, similarly to the work of Zaremba at al. [36]. Accordingly, the LSTMs attempt to predict the next step in a stochastic manner. The first LSTM layer creates a feature vector while the second calculates the final predictions. In each prediction for time index n, the network outputs the probabilities for every state. The feed-forward layer has 256 hidden units and its output is fed into merge layers, along with the output of the LSTM blocks. Finally, each merge layer then passes through the *softmax* nonlinearity to generate the probability distribution of the prediction. During the optimisation phase, the Adam [21] algorithm is used while the loss function is calculated as the mean of the (categorical) cross-entropy between the prediction and the target of each output. For our experiments, we used the Tensorflow [1] deep learning framework and Keras [6] library.

4 Results

The conditional (feed-forward) part of the CNSL architecture allows the inclusion of additional information or "constraints" that potentially keep the entire network "on-track" and enable it to respond to conditions that were not encountered during training (e.g. unknown—to the network—time signatures). Therefore, training and generating with a network that does not incorporate the conditional part is expected to lead to generated rhythms that could not possibly adapt to, e.g. time signature changes. While different networks without conditional layers can be used on parts with different time signatures, e.g. a network trained in 4/4 for the 4/4 parts and one trained on 5/4 for the 5/4 parts, the conditional layer offers another important advantage: it allows the generation of drums' sequences in time signatures that *have not been used during training*. For instance, even if a CNSL network has not encountered a 17/16 time signature during training, the metric information provided in the conditional layer (which incorporates encoded information that corresponds to the accentuations of a 17/16 time signature) will allow the network to understand the previously unseen metric structure and adapt to it.

The aim of the presented results is to validate the role of the conditional layer and to highlight potential weaknesses in the ability of the network to generate drums' rhythms. The role of the conditional layer is examined by testing



(a) Architecture 1 **(b)** Architecture 2

Fig. 1 Proposed deep neural network architectures for conditional drums generation

both proposed architectures trained on separate bands with diverse music characteristics. Furthermore, different approaches regarding the inclusion of selected features of the conditional layer are presented in specific trained networks.

4.1 Experimental setup

Music from bands has been included in the training corpus that belongs to different styles, each one having unique rhythmical characteristics. Regarding drums' compositions, *AB* (pop/disco band) songs are easily recognised with the continuous hi-hat playing in high tempos. On the other hand, *PF* (blues/rock band) is identified with the slow tempo and simple patterns, especially with Ride cymbals. Finally, *PT* (progressive/metal band) is characterised by the complex playing, even in 4/4 time signatures, along with continuous changes in tempo and time signature. It is worth noting that *PT* is considered as the evolution of *PF* in the music industry. However, a significant difference can indeed be found between the repetitive drum playing of *PF* band and the constantly changing rhythms of the *PT* band.

Two separate experiments are conducted to highlight different aspects of the conditional layer.

(a) Genre music style capture: The aim of this experiment is to validate the CNSL architecture for generating drums' rhythms that resemble the learnt style for each corresponding band of the training corpus. $\texttt{Arch}_1$ and $\texttt{Arch}_2$ are employed as core components of the LSTM part of the network (described in Sect. 3.1.2).

(b) Examining the conditional layer: several approaches concerning the impact of the feature selection (described in Sect. 3.2) of the conditional layer are presented.

4.2 Validating CNSL architecture for capturing certain musical styles

In order to validate our system, we use nine pieces (three for each band) which were not included in the training dataset with the following music parameters:

1. AB_1 with sparse time signature changes (2/4 and 4/4) and high tempo
2. AB_2 with 4/4 time signature and high tempo
3. AB_3 with 4/4 time signature and very high tempo
4. PF_1 with 4/4 time signature and low tempo
5. PF_2 with 4/4 time signature and very low tempo
6. PF_3 with 4/4 time signature and moderate tempo

Table 2 Style-specific composition assessment: networks trained in the style of *AB*, *PF* and *PT*, in both architecture variations $\texttt{Arch}_1$ and $\texttt{Arch}_2$, are employed to compose drums' sequences using nine test tracks

	$\texttt{AB}_{\texttt{Arch}_2}$	$\texttt{PF}_{\texttt{Arch}_2}$	$\texttt{PT}_{\texttt{Arch}_2}$	$\texttt{AB}_{\texttt{Arch}_1}$	$\texttt{PF}_{\texttt{Arch}_1}$	$\texttt{PT}_{\texttt{Arch}_1}$
$\texttt{AB}_1$	**0.48** (0.05)	0.57 (0.01)	0.66 (0.24)	**0.40** (0.08)	0.70 (0.08)	0.75 (0.10)
$\texttt{AB}_2$	**0.42** (0.16)	0.52 (0.01)	0.64 (0.04)	**0.51** (0.05)	0.62 (0.01)	0.54 (0.06)
$\texttt{AB}_3$	0.38 (0.10)	**0.33** (0.02)	0.44 (0.08)	**0.23** (0.05)	0.39 (0.01)	0.51 (0.03)
$\texttt{PF}_1$	0.73 (0.03)	**0.21** (0.01)	0.39 (0.09)	0.76 (0.01)	**0.23** (0.03)	0.39 (0.10)
$\texttt{PF}_2$	0.70 (0.03)	**0.29** (0.05)	0.31 (0.02)	0.49 (0.03)	0.46 (0.03)	**0.24** (0.01)
$\texttt{PF}_3$	**0.17** (0.03)	0.19 (0.01)	0.25 (0.01)	0.22 (0.04)	**0.22** (0.02)	0.38 (0.04)
$\texttt{PT}_1$	0.29 (0.04)	0.31 (0.03)	**0.09** (0.01)	0.40 (0.00)	0.36 (0.10)	**0.10** (0.09)
$\texttt{PT}_2$	0.21 (0.04)	0.36 (0.03)	**0.09** (0.02)	0.26 (0.03)	0.37 (0.09)	**0.04** (0.02)
$\texttt{PT}_3$	0.44 (0.06)	0.29 (0.02)	**0.02** (0.02)	0.62 (0.13)	0.42 (0.05)	**0.00** (0.00)

Mean values and standard deviations (in parentheses) of distances between ground-truth (original drums) and system-generated drums features are given; the minimum distances for each piece (row) in each architecture (left/right column triplets) are shown in bold

7. PT_1 with 7/8 time signature and moderate tempo
8. PT_2 with continuous time signature changes (4/4, 3/8, 5/8 and 7/8) and high tempo
9. PT_3 with sparse time signature changes (17/16, and 4/4) and low tempo

The proposed CNSL architecture is able to generate sequences that reflect characteristics of the training data; however, one can suspect that the conditional layer might have no impact, or even to jeopardise the ability of the LSTM to generate distinguishably different music styles. To this end, networks trained in the style of *AB*, *PF* and *PT*, in both architecture variations $\texttt{Arch}_1$ and $\texttt{Arch}_2$, are employed to compose drums' sequences using the nine test tracks that were presented above. To validate the effectiveness of the conditional layer, drums' rhythms generated by systems trained in a certain style are compared with the same drums' rhythms generated from systems trained in different learnt styles in order to examine the ability of a network to imitate a learnt style. For example, if the system is trained with the *PT* dataset and it is "assigned" (i.e. given conditional constraints) to compose pieces from *PT* (not encountered during training), will it do it "*better*" than when it is assigned to compose pieces from the *AB* style?

Considering that ground-truth for training and testings pieces is available (the pieces as originally composed), we can approach the notion of "better resemblance" by measuring how close the system-generated compositions are to the ground-truth; this can be done by computing the distance between cognitively relevant feature representations of the drums' rhythms in these pieces. Such features have been employed for successfully classifying drums' rhythms according to style [18], and for generating rhythms with genetic algorithms [19]. Thereby, taking the first 16 bars[4] from original and generated drums, features are extracted which estimate numerical values regarding density, syncopation [32], symmetry and weak-over-strong onset ratio [19] concerning basic rhythmic elements: kick, snare/toms and hi-hat/cymbals; 12 features (4 features for each of the three percussive categories) for each bar are extracted. The feature values in each bar are summarised with mean and standard deviation, producing a set of 24 global features for every piece.

Table 2 shows the aforementioned distances, using the proposed architectures to learn the corresponding styles. The diagonal blocks in both parts of the matrix, include mostly the lowest values (indicated as bold numbers). The results show that style is generally reflected by the networks. Especially in the case of *PT*, which features the most idiosyncratic characteristics, both networks appear to have learnt this style better, since the generated rhythms from the PT dataset appear to be significantly closer to the originally composed rhythms.

4.2.1 Examples of generated rhythms

A major difference can be identified between the two aforementioned architectures, specifically by listening to some generations. $\texttt{Arch}_1$ seems to create less complex drum rhythm generations than $\texttt{Arch}_2$, especially in occasions where rare (in the context of training) time signatures are incorporated. Music examples with generated rhythms are available on the web page of Humanistic and Social Informatics Lab of Ionian University, Greece.[5]

Figure 2 illustrates three examples of generated drums' rhythms, corresponding to different learning settings for the first verse (phrase) of the PT_3 song, showing 2 consecutive bars. Starting from bottom to top, in the piano-roll drum depiction, one may identify in the following order

[4] Features are not extracted from the entire piece to make sure that the results are not influenced by the length of the pieces; longer pieces move further away from the seed sentence, potentially accumulating more errors during their life-span.

[5] https://hilab.di.ionio.gr/en/english-conditional-neural-sequence-learners-for-generating-drums-rhythms/.

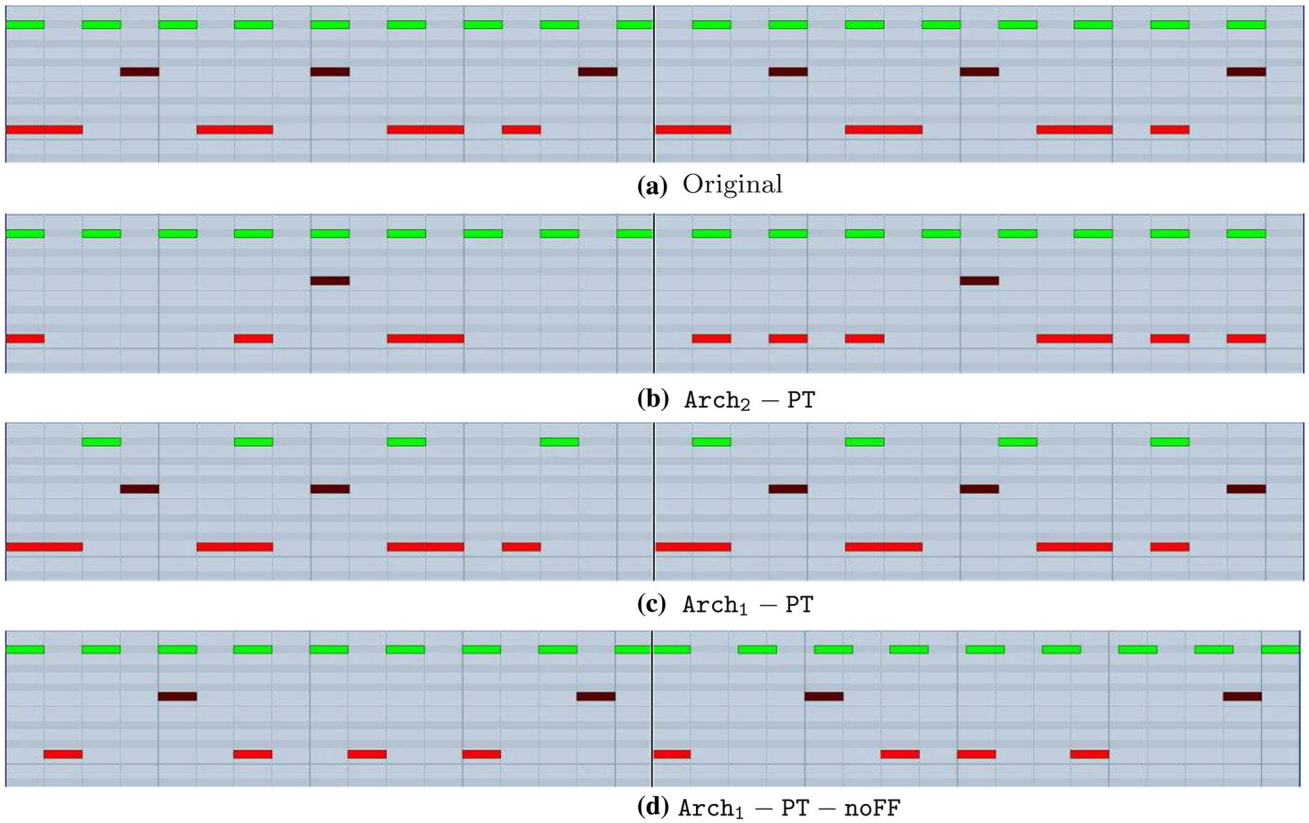


Fig. 2 Pianoroll demonstration of 2 bars with time signature 17/16 of PT_3 song on **a** the original drums, **b** drums composed from $\texttt{Arch}_1$ in the learnt style of PT, **c** $\texttt{Arch}_2$ with style PT, and **d** $\texttt{Arch}_2, \texttt{PT}$ style, without the conditional layer

kick (red), snare (brown), ride (blue), hi-hat (green), and crash (pink) events. This example was selected in order to observe how the system reacts on time signatures that have not been learnt from the training data and the importance of the conditional (FF) layer. According to the ground-truth, there is a pattern on kick-snare and hi-hat events in specific beats for the 17/16 time signature which both architectures with the conditional layer capture efficiently. On the other hand, experiments without the conditional layer are failing to adapt the particular musical characteristics and generate patterns which have been encountered more often on the training data leading to confusing rhythms' generations that misalign with the metric structure and the events of other instruments.

On the other hand, the second example shows notable approaches of PT_2 song from different trained networks. Figure 3 features 3 bars' generation with time signatures 5/8 and 7/8, from PF and PT networks using $\texttt{Arch}_2$ architecture. It is thus clear that the PT network captures the corresponding style and it is closer to the original. However, the PF network, even if it is unaware of the above time signatures (PF dataset is almost consisted of songs with 4/4 time signature), follows the same rhythm with "less" playing in terms of density. This is caused by the conditional layer which forces the LSTM layers to generate in a specific rhythm that has not been encountered in the training process, in the learnt style, which in this case leads to less complex drums than the PT network.

4.3 The role of the conditional layer: a study on composite beat profiles

The conditional layer employs information regarding the metrical structure, tempo and phrasing mapping, as well as the accompanying instrumentation (bass and guitar) on the musical surface. The importance of each element in the conditional layer can be examined through measuring the periodicity of repeating onsets of kick, snare, hi-hat and ride, toms and cymbals hits within the rhythm in specific beats of the measures throughout the piece. The overall characteristics of the drums' rhythms used in a piece can be observed by the *composite beat profile*, which reflects the distribution of all elements of the drums' rhythms of all measure of the same time signature in a piece. Composite beat profile is defined as an array that describes the frequency counts of each drum's onset on each beat of the measure, forming a distribution on each beat position (according to the metric resolution). The idea of employing frequency count of musical events for inducing the metric

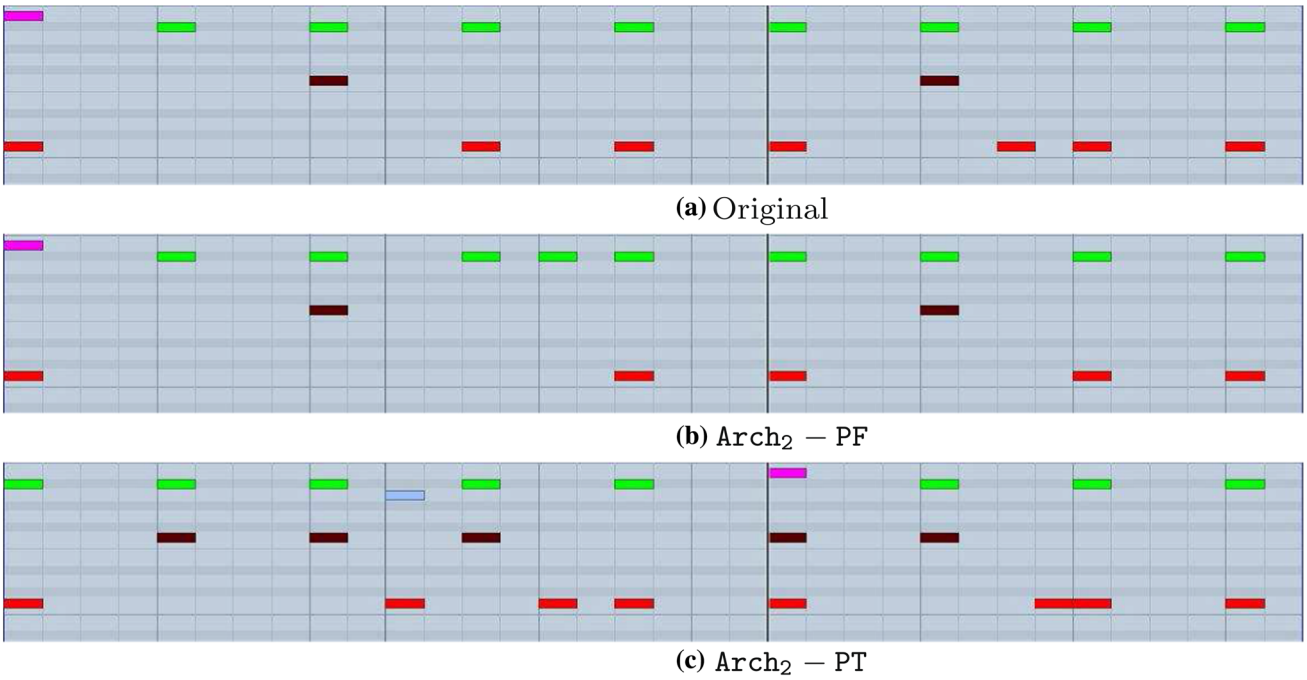


Fig. 3 Pianoroll demonstration featuring 3 bars with time signatures 5/8 and 7/8 of PT_2 song on **a** the original drums, **b** drums composed with $\texttt{Arch}_2$ in the style of PF, and **c** $\texttt{Arch}_2$ in the style of PT

structure was discussed as far back as in 1990, in the work presented by Palmer and Krumhansl [30].

Composite beat profiles of drums' rhythms composed by $\texttt{PT}_{\texttt{Arch}_1}$ network are depicted in single bar plots, where different colours represent different drums' elements at each position in the measure. For our experiments we used four different setups regarding the feature selection of the conditional layer:

1. FF − Full—metrical, tempo, phrasing and instrumentation information included.
2. FF − M—metrical, tempo, phrasing information included.
3. FF − I—instrumentation information included.
4. FF − None—conditional layer is not employed.

The example of Fig. 4 demonstrates the corresponding generations of the PT_3 piece in time signature 17/16, which has not been encountered during training. Figure 4a shows the composite beat profile of the "ground-truth" piece, i.e. the drums that are actually included in the genuine composition; the basic structural component in this figure appears to be the strong presence of kicks and crashes in the first beat (shown as beat 0). Moreover, kicks are encountered in bars 1, 5, 6, 10, 11 and 13. When the conditional layer is not used (Fig. 4b) irregular drum patterns with low density of kicks, and high presence of hi-hats are spread all over the beats. This is caused by the absence of the conditional layer, which shows the weakness of the LSTM layers to generate without constraints. Figure 4c has almost the same results with the FF − None setup, thus leading us to the conclusion that the metrical information is very crucial in such complex tasks. On the other hand, FF − M manages to give better results than the previous setups; however, it fails on some beats about the presence of the kicks. If we combine metrical and instrumentation information, Fig. 4e shows that the FF − Full manages to capture efficiently the structure of an unknown for the system 17/16 time signature. Although it fails on the density of hi-hats, the kicks' presence is almost similar with the ground-truth.

Figure 5 shows an example of a low tempo song with sparse drumming in time signature of 7/8. The PT_1 piece has a strong repeated pattern in beats of every measure, which is also followed by the same playing pattern of the accompanying instruments (bass and guitar). Figure 5a shows the composite beat profile of the "ground-truth" piece. The basic structural component in this figure appears to be the strong presence of kicks in beats 1 and 3 while snare is only encountered in beat 9. When the conditional layer is not used (FF − None setup) kicks' and snares' occurrences are detected in every odd beat, thus leading to a completely different pattern. Figure 4d shows that FF − I, with some minor mistakes, captures the style of the drums' pattern. However, and mostly remarkable fact of the above case is the generation that FF − M setup achieves. Figure 4c is almost identical with the ground-truth and manages to achieve slightly better results than using all the features of the conditional layer (FF − Full). This can be

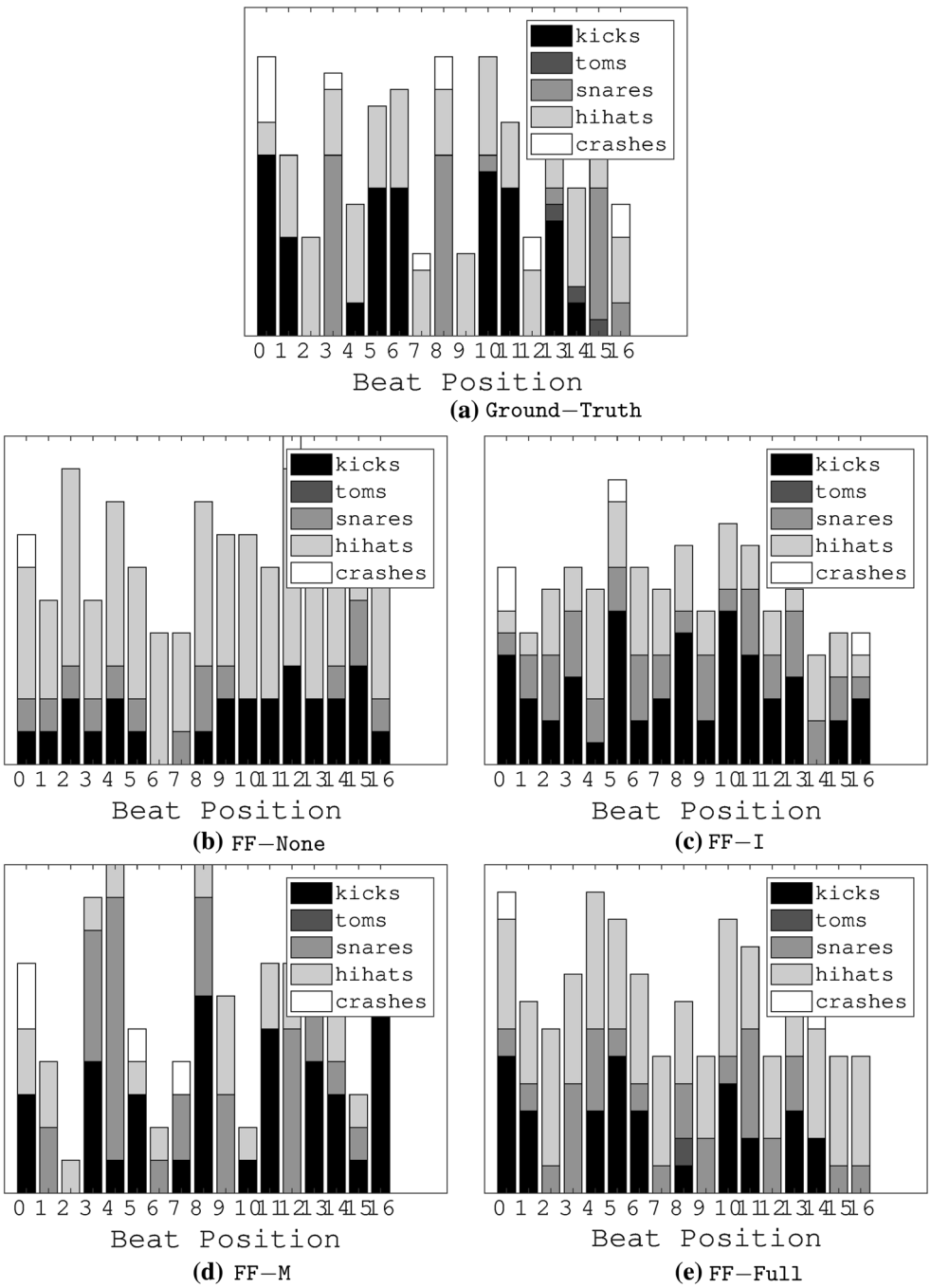


Fig. 4 Composite beat profiles of **a** the original drums of the piece PT_3 and the drums' composition using the following setups: **b** `FF − None`; **c** `FF − I`; **d** `FF − M`; and **e** `FF − Full`

explained by the important role of the instruments accompanying a human drummer and how these affect the production of drums' patterns.

5 Conclusions

This work introduces a novel neural network architecture, namely the conditional neural sequence learner (CNSL), which learns and generates sequences under given constraints; an application on learning and generating drums rhythms under given metrical, bass and guitar constraints is presented. The proposed architecture consists of a recurrent module with LSTM blocks that learns sequences of consecutive drums' events along with a conditional (feed-forward) layer handling information for musical instruments, metrical structure, tempo and the grouping (phrasing).

Two variations of the proposed architecture are tested, with different settings about the learnt style and the feature

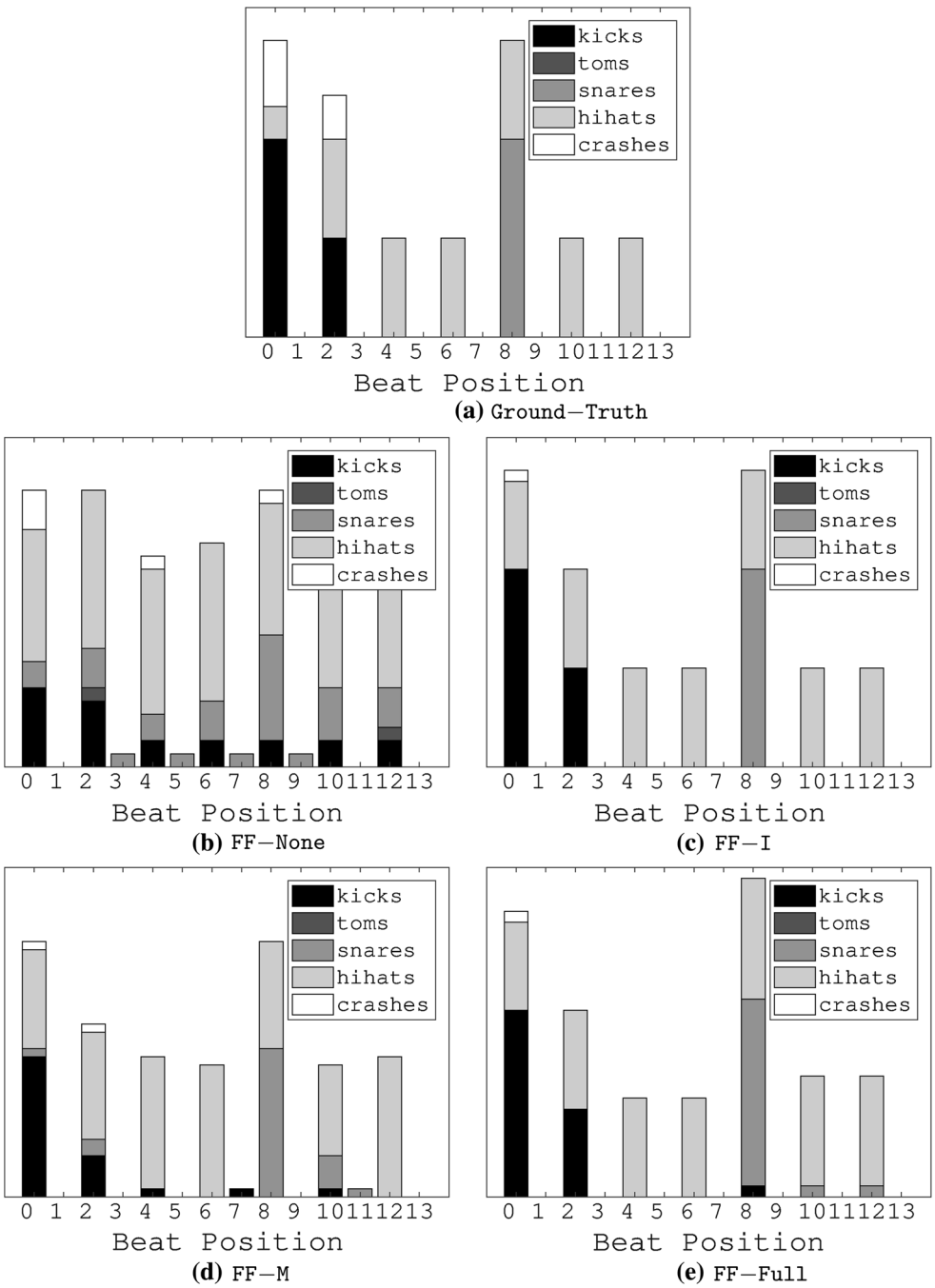


Fig. 5 Composite beat profiles of **a** the original drums of the piece PT_1 and the drums' composition using the following setups: **b** FF − None; **c** FF − I; **d** FF − M; and **e** FF − Full

selection of the conditional layer. The experimental setup includes test songs from three different datasets with diverse musical characteristics regarding time signatures and tempo changes; distances between generated and original drums' rhythms were calculated, while composite beat profiles, respectively were compared highlighting certain aspects of the conditional layer.

Results indicated that CNSL architecture is able of producing generated drums' rhythms that resemble a learnt style, while the comparison indicated that the conditional layer is necessary for allowing the network to follow changing, rare or even previously unseen (e.g. 17/16) time signatures. Additionally, the inclusion of musical information about grouping, tempo and instrument playing simulate human drummers in tasks such us: changing the playing style when a new phrase starts, responding to tempo changes influenced by the density of the drums' generation and responding to changes in other instruments (e.g. guitar, bass).

Our future work will include the extension of the available training data, along with the expansion of network architecture with additional elements for the conditional layer featuring information of other instruments and meta-data. Furthermore, we will set the quantisation to 1/64 to include triplet representations which are very commonly used on drums' rhythms. Finally, more experiments will be conducted to test the system's behaviour in different music genre styles and its ability to adapt to the performance of human drums' players.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M (2016) Tensorflow: a system for large-scale machine learning. OSDI 16:265–283
2. Allan M, Williams C (2005) Harmonising chorales by probabilistic inference. In: Advances in neural information processing systems, pp 25–32
3. Briot JP, Hadjeres G, Pachet F (2017) Deep learning techniques for music generation—a survey. arXiv preprint arXiv:1709.01620
4. Cardoso A, Veale T, Wiggins GA (2009) Converging on the divergent: the history (and future) of the international joint workshops in computational creativity. AI Mag 30(3):15
5. Choi K, Fazekas G, Sandler M (2016) Text-based LSTM networks for automatic music composition. arXiv preprint arXiv:1604.05358
6. Chollet F et al (2015) Keras
7. Cunha US, Ramalho G (1999) An intelligent hybrid model for chord prediction. Organ Sound 4(2):115–119
8. Deliège I, Wiggins GA (2006) Musical creativity: multidisciplinary research in theory and practice. Psychology Press, London
9. Eck D, Schmidhuber J. (2002). A first look at music composition using LSTM recurrent neural networks. Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale, vol 103
10. Edwards M (2011) Algorithmic composition: computational thinking in music. Commun ACM 54(7):58–67
11. Fernández JD, Vico F (2013) AI methods in algorithmic composition: a comprehensive survey. J Artif Intell Res 48:513–582
12. Hadjeres G, Pachet F (2016) Deepbach: a steerable model for Bach Chorales generation. arXiv preprint arXiv:1612.01010
13. Hild H, Feulner J, Menzel W (1992) Harmonet: a neural net for harmonizing Chorales in the style of J. S. Bach. In: Advances in neural information processing systems, pp 267–274
14. Hiller LA, Isaacson LM (1959) Experimental music; composition with an electronic computer. McGraw-Hill, London
15. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
16. Hutchings P (2017) Talking drums: generating drum grooves with neural networks. arXiv preprint arXiv:1706.09558
17. Jacob BL (1996) Algorithmic composition as a model of creativity. Organ Sound 1(03):157–165
18. Kaliakatsos-Papakostas M (2018) Generating drum rhythms through data-driven conceptual blending of features and genetic algorithms. In: International conference on computational intelligence in music, sound, art and design. Springer, Berlin, pp 145–160
19. Kaliakatsos-Papakostas M, Floros A, Vrahatis MN (2013) Evodrummer: deriving rhythmic patterns through interactive genetic algorithms. In: International conference on evolutionary and biologically inspired music and art. Springer, Berlin, pp 25–36
20. Kalingeri V, Grandhe S (2016) Music generation with deep learning. arXiv preprint arXiv:1612.04928
21. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
22. Leach J, Fitch J (1995) Nature, music, and algorithmic composition. Comput Music J 19(2):23–33
23. Lewis J (1989) Algorithms for music composition by neural nets: improved CBR paradigms. Michigan Publishing, University of Michigan Library, Ann Arbor
24. Liang F (2016) Bachbot: automatic composition in the style of Bach Chorales. PhD thesis, Masters thesis, University of Cambridge
25. Makris D, Kaliakatsos-Papakostas M, Karydis I, Kermanidis KL (2017) Combining LSTM and feed forward neural networks for conditional rhythm composition. In: International conference on engineering applications of neural networks. Springer, Berlin, pp 570–582
26. Merker BH (2006) Layered constraints on the multiple creativities of music. In: Musical creativity: multidisciplinary research in theory and practice, pp 25–41
27. Mozer MC (1999) Neural network music composition by prediction: exploring the benefits of psychoacoustic constraints and multiscale processing. In: Musical networks: parallel distributed perception and performance, vol 227
28. Nierhaus G (2009) Algorithmic composition: paradigms of automated music generation. Springer, Berlin
29. Pachet F, Roy P (2001) Musical harmonization with constraints: a survey. Constraints 6(1):7–19
30. Palmer C, Krumhansl CL (1990) Mental representations for musical meter. J Exp Psychol Hum Percept Perform 16(4):728–741
31. Papadopoulos G, Wiggins G (1999) AI methods for algorithmic composition: a survey, a critical view and future prospects. In: AISB symposium on musical creativity. Edinburgh, UK, pp 110–117
32. Sioros G, Guedes C (2011) Complexity driven recombination of midi loops. In: Proceedings of the 12th international society for music information retrieval conference (ISMIR). University of Miami, Miami, USA, pp 381–386
33. Sturm B, Santos JaF, Korshunova I (2015) Folk music style modelling by recurrent neural networks with long short term memory units. In: 16th international society for music information retrieval conference (ISMIR)
34. Supper M (2001) A few remarks on algorithmic composition. Comput Music J 25(1):48–53
35. Wiggins GA, Pearce MT, Müllensiefen D et al (2009) Computational modeling of music cognition and musical creativity. Oxford University Press, Oxford
36. Zaremba W, Sutskever I, Vinyals O (2014) Recurrent neural network regularization. arXiv preprint arXiv:1409.2329