Towards an Evaluation of Symbolic Music Encodings for RNN Music Generation

Manos Plitsis[1], Kosmas Kritsis[2,3], Maximos Kaliakatsos-Papakostas[2], Aggelos Pikrakis[2,3] and Vassilis Katsouros[2]

[1] Dept. of Computer Science, Sorbonne University, France
{plitsis}@ircam.fr
[2] Institute for Language and Speech Processing, Athena R.C., Greece
{kosmas.kritsis, maximos, vsk}@athenarc.gr
[3] Dept. of Informatics, University of Piraeus, Greece
{pikrakis}@unipi.gr

Abstract. The choice of encoding for symbolic music data used in music generation models has typically been done in a case-by-case basis up to now. In this paper we attempt to evaluate and study the behaviour of a baseline shallow LSTM network trained on Irish folk music, using three different encodings: a MIDI-like event-based encoding, a fixed timestep-based encoding, and the popular ABC notation. We use objective statistical measures on the network output and also visualise the LSTM parameters to gain insight on the way it processes each encoding.

Keywords: Music generation, Symbolic Music Encoding, LSTM, Recurrent Neural Networks

1 Introduction

Symbolic music generation with Neural Networks (NNs) is a sub-field of Algorithmic Composition and Artificial Intelligence (AI) that, while fairly old, has not seen the advancements of fields like Natural Language Processing, while borrowing many of its methods. This partly has to do with the abstract, semantically ambiguous nature of music, compared to natural language. In our opinion, this is also due to the choice of representation for musical data, which has not been standardized, and is usually modelled after transcription methods intended for human readability (e.g. ABC notation, tablature), or popular standards whose function was not designed for this use (e.g. MIDI). While these issues have been addressed in the past (Dannenberg, 1993; Byrd, Boyle, Berggren, Bainbridge, et al., 1997; Honing, 1993), there have been to the best of our knowledge no studies in the empirical effects of encoding using specific NN generation systems.

In this paper we limit our scope to sequence-based representations where steps are musically meaningful, in order to cover popular encodings like ABC, piano-roll and MIDI-like encodings. We view the choice of encoding not just as another hyperparameter for a network or other statistical model that is arbitrarily chosen in practice, but as an integral part of a system, the choice of which is informed by a theoretical and empirical understanding of its effects.

2 Encodings

As our original dataset, we use a collection of Irish folk tunes, transcribed in the ABC music format, which was collected and preprocessed by the *folk-rnn* team (B. L. Sturm, Santos, Ben-Tal, & Korshunova, 2016) [4], and on which *folk-rnn* has been trained on. From there we extract two more encodings, one based loosely on the MIDI protocol which we denote *event1* and another based on fixed timestep representations which we denote *tstep1*. Here we give their basic specifications and individual information.

In **tstep1**, each piece is represented as an array of integer tokens in the range 0 – 129, where: tokens 0 – 127 denote a note-on event, 128 denotes a rest and 129 means "continue playing the last-seen note".

To encode a piece of music, we first specify a resolution parameter, which gives the length of the timestep in subdivisions of a quarter note: a resolution of 4 means the timestep length is 1/16th. As an example, when transcribing a C3 quarter note followed by two C3 eighth notes, we get the sequence [60, 129, 129, 129, 60, 129, 60, 129]. Obviously, the smallest note length that can be represented is equal to the timestep length, so all pieces are quantized during conversion. Timestep-based encodings were used extensively from the early applications of NNs to music generation (Todd, 1989; Eck & Schmidhuber, 2002), as it was believed that having a fixed timestep helped a model to better process rhythm (Gers & Schmidhuber, 2000).

event1 is loosely based on the MIDI protocol, containing events for note-on and note-off, while the passage of time is explicitly notated with special "move forward in time" events. Specifically: tokens 0 –127 denote note-on events, 128 – 255 denote note-off events, 256 – 356 denote time-shift events.

Time-shift events move time forward by some increment, which herein is based on a grid, but can also be in seconds. In our implementation, 256 denotes the shortest time interval used (equivalent to a 32nd note), and subsequent events denote its multiples. This facilitates the comparison with timestep-based encodings. A typical pattern, especially with well-quantized pieces is: note-on, wait, note-off(, wait), note-on etc. So for each note in the original piece we would need three or four distinct tokens to encode it, meaning that there is a strict bound to the events needed to represent a piece, which does not grow with resolution. For most music, this is significantly less than any timestep-based encoding. This is especially important when the goal is to capture real-time performance, where the step duration can be as low as 1/44100 of a second. This encoding, with the addition of velocity events, was used in Magenta's Performance RNN (Oore, Simon, Dieleman, Eck, & Simonyan, 2020).

As **ABC notation** is an ever-evolving transcription language, there is no definitive, formal "ABC standard"[5]. As the "pure" text-based ABC format is, strictly speaking, outside the scope of this paper, we employ the "tokenized"

[4] Specifically data_v3

[5] The reader is advised to reference the latest documentation on the abc standard `http://abcnotation.com/wiki/abc:standard`

version (also used by *folk-rnn*), which transforms it by grouping symbols together to create *musically meaningful* symbols. This *tokenization* is arbitrary: a different choice would lead to another distinct encoding, e.g. the duration of a note could be included in note-tokens or the note octave could be a separate token. Header information serves as a sort of classifying token, when tunes of different meter (M), key (K) or default note length (L) are in the same dataset. In our experiments, we further filtered the *folk-rnn* dataset pieces to keep only those in the key of C or its modes with a 4/4 time signature, and discarded all header information, adding only start and stop symbols at the beginning and end of songs. We kept only the 4/4 tracks so that we have a uniform dataset and we can later detect how well each encoding translates metric information. The reason for removing headers is that the other two encodings have implicit metric and key information, while the ABC header can act as a classifier, giving extra information to the system. By having no header information we can autoregressively generate pieces of arbitrary length. This resulted to a total of 12117 transcriptions.

3 Empirical Evaluation

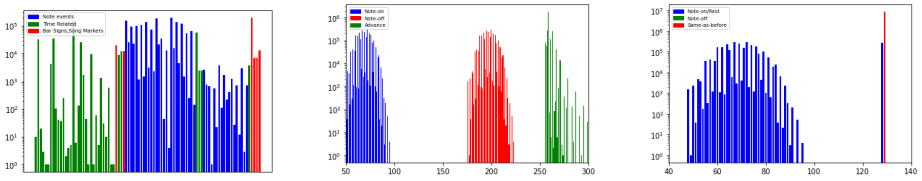
Our aim is to use music-specific objective statistical measures to assess the network's output similarity to the original dataset, as in (Yang & Lerch, 2018). Another objective is to inspect the model's parameters to study how it represents music internally, similarly to (Karpathy, Johnson, & Fei-Fei, 2015) and (B. Sturm, 2018).

3.1 Experimental Setup

Dataset Starting from the dataset described in Section 2, we convert the ABC pieces to MIDI and use them to extract the event1 and tstep1 encoded datasets, which consist of a one-dimensional integer vector for each piece, with each integer representing an event as described in Section 2. We use a dictionary with length equal to the number of different events in the dataset to further encode each piece. We split each dataset into a training and validation set (the validation set being 10% of all pieces), each containing the same pieces across datasets.

The resulting datasets are different in many ways. The sequence length needed to represent one measure is 8.71 (STD 3.362) for the ABC dataset, 20 (STD 3.939) for the event-based encoding, and always 32 for the fixed timestep one. The token distribution is also very different: in Figure 1 we plot (in log-scale) the frequency of each token for the three versions of the dataset. Notice how skewed the tstep1 distribution is, or the prevalence of "advance-by-1/8" (the spike in the green area of the event1 plot).

Network Architecture and Training We choose a small, shallow LSTM network with one layer and a hidden state of dimension 32, followed by a Dense layer for each output of dimension equal to the vocabulary size. The Dense layer



(a) ABC (b) tstep1 (c) event1

Fig. 1: Distribution of distinct tokens for each representation (left to right: abc - event1 - tstep1). The y axis denotes absolute number of occurrences in a logarithmic scale.

outputs are then passed by a softmax function to yield a probability distribution for each timestep, from which we sample the dictionary index for the next prediction during inference. We train the network with partially overlapping sequences of 100 one-hot encoded vectors, using a sliding window across the length of each piece. The categorical crossentropy loss is computed for batches of size 256, between each input sequence and the sequence of outputs of the Dense layer and the network gradients are back-propagated at the end of each batch. The weights are updated using the Adam optimizer, with an initial learning rate of 0.001. During training we monitor the validation loss, stopping the training when it starts increasing for more than two epochs.

3.2 Results

We generate 200 pieces from each model that was trained with a different encoding. We use as seeds the beginning of the first 200 tunes from the validation dataset (1205 pieces long), with variable seed length so as to represent exactly one bar. We generate exactly 8 bars for each song to obtain 200 9-bar samples for each encoding plus 200 9-bar samples from the original dataset. The generated pieces can be found in the followig link: `https://github.com/manosplitsis/MusicRep/tree/master/JCIM2020_listening_data`. We process the samples according to (Yang & Lerch, 2018): First, music related statistical measures are computed for each set. An exhaustive cross validation inside each dataset (inter-set) and between pairs of datasets (intra-set) is computed measuring the euclidean distance of each measure. These relative measures give a histogram for each feature, from which we compute a continuous Probability Density Function (PDF) using Kernel Density Estimation. To compare the generated data with the original, we compute the KL Divergence (KLD) and Overlapping Area (OA) between the intra-set PDF of the generated data with the inter-set PDF between the generated and original data. A low KLD indicates similarity in the shape of compared distributions while high OA indicates higher probability density overlap. This will allow us to compare how closely related is each model's output to the original dataset with respect to each measure. We were not able to find any major distinction in any measurement using this method. All three models seem

to adequately model the dataset's characteristics, as seen by the small KL divergence and fairly high OA (tables of the results can be seen in the Appendix, in Table 2). This contradicts our experience when listening to the generated pieces, that vary in quality and possess different characteristics. Specifically, the ABC trained network gives overall the best results, producing seemingly human-like musical pieces with some kind of structure (this is facilitated by the existence of repeat barlines). The *event1* trained network gives fairly good results melodically and structurally, but contains a lot of syncopation which is not present in the original dataset. An advance time token is predicted after a note-on token, but its duration is often unusual, possibly due to randomness from sampling. The *event2* trained network output is the weakest of all three, something which is consistent with earlier experiments (Plitsis, Kritsis, Kaliakatsos-Papakostas, & Katsouros, 2020). This suggests that the selected measures may not be well-suited to the task.

One thing shared more or less by all the networks' outputs (despite all being mainly in the right key) is their ability to consistently output music that sounds like it is in 4/4. We decided to visualise the activations for each of the 32 neurons of the LSTM layer for the beginning of a tune in all encodings, to see if we can correlate it to metric (or other) information. Some things immediately become evident: for ABC notation neurons 9 and 22 seem to be responsible for keeping track of barlines, with neuron 9 activating when a barline is fed to the network and 22 anticipating it (see the high probability of a barline occuring after activations). In the other encodings (that do not have explicit barlines) we find no apparent way to tell if the network knows how to count. On the other hand some local patterns become clearer: In event1, after a note-on event the network is fairly sure to output an advance-time event (specifically one equivalent to 1/8th, the default note-length for all pieces in the original ABC dataset). Note-on events have fewer activations (specifically one neuron is consistently activated) and are not biased towards one particular prediction, which means that the output is quite variable melodically. The three network parameter visualisations are presented in the Appendix A.2.

4 Conclusions and Future Work

Despite its limitations, we believe that an extended research on the effects of encoding in musical data, is essential to the field of music generation and music information retrieval. On the one hand, it can help with analyzing the behaviour of specific network architectures, when confronted with the same semantic information encoded in different ways. On the other hand, it can aid in the design of new musical encodings, better suited for a system's task, whatever it might be. Future work involves a better understanding of the effects of re-encoding musical information, the use of state-of-the-art architectures such as Transformers, the comparison with natural language generation systems and the way they encode language information, as well as the design of standardized benchmarks for large-scale experiments with many different encodings.

A Appendix

A.1 Objective Measurements of the network output

Table 1: Absolute and intra-set differences for all measures Measure abbreviations: Pitch Count(PC), Note Count (NC), Pitch Class Histogram (PCH), Pitch Class Transition Matrix (PCTM), Pitch Range (PR), Average Pitch Shift (PS), Average Inter-onset Interval (IOI), Note Length Histogram (NLH), Note Length Transition Matrix (NLTM).

	Dataset				tstep1				abc				event1			
	Abs.Value		Intra-Set		Abs.Value		Intra-Set		Abs.Value		Intra-Set		Abs.Value		Intra-Set	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
PC	9.590	1.689	1.906	1.449	13.075	2.371	2.606	2.122	11.720	2.283	2.544	2.001	12.530	12.530	2.294	1.804
PC/bar			4.528	1.772			4.770	1.298			4.709	1.475			4.575	1.180
NC	61.520	10.112	10.50	9.761	62.810	6.621	6.918	6.345	62.785	7.782	8.277	7.295	58.345	58.345	6.168	5.632
NC/bar			4.953	2.973			5.367	2.155			5.315	2.350			4.919	1.894
PCH			0.301	0.103			0.191	0.059			0.233	0.079			0.191	0.060
PCH/bar			0.093	0.376			0.062	0.307			0.732	0.789			1.492	0.186
PCTM			16.933	3.674			11.225	1.623			13.182	2.551			10.855	1.684
PR	15.810	3.208	3.595	2.787	21.365	4.143	4.597	3.656	19.455	4.370	4.955	3.719	20.220	20.220	4.044	3.095
PS	2.967	0.615	0.669	0.560	2.853	0.341	0.379	0.301	2.908	0.453	0.501	0.402	2.827	2.827	0.404	0.323
IOI	0.302	0.061	0.056	0.066	0.290	0.039	0.036	0.041	0.285	0.038	0.037	0.039	0.294	0.294	0.038	0.040
NLH			0.376	0.336			0.287	0.282			0.328	0.323			0.299	0.310
NLTM			24.677	17.009			20.136	15.491			22.569	17.336			18.892	15.395

Table 2: Similarity statistics between dataset and generated distributions. Measure abbreviations are explained in the Table 1 caption.

		PC	PC/bar	NC	NC/bar	PCH	PCH/bar	PCTM	PR	PS	IOI	NLH	NLTM
tstep	KLD	.1369	.2657	.0078	.0924	.0080	.1686	.0456	.0718	.0141	.0650	.0230	.0133
	OA	.7419	.9351	.8675	.8942	.6580	.4799	.4780	.7893	.8306	.8649	.8586	.9067
abc	KLD	.1746	.1046	.0088	.0334	.0234	.1947	.1120	.0075	.0043	.1280	.0151	.0129
	OA	.8775	.9549	.9062	.8981	.8251	.5556	.7153	.9267	.9048	.8709	.9074	.9314
event	KLD	.2605	.5395	.0395	.3559	.0552	.9893	.4550	.0296	.0085	.3785	.0397	.0194
	OA	.7358	.9366	.7323	.9041	.6768	.7352	.4495	.8132	.8464	.8615	.8529	.8536

A.2 Visualisation of the Network activations

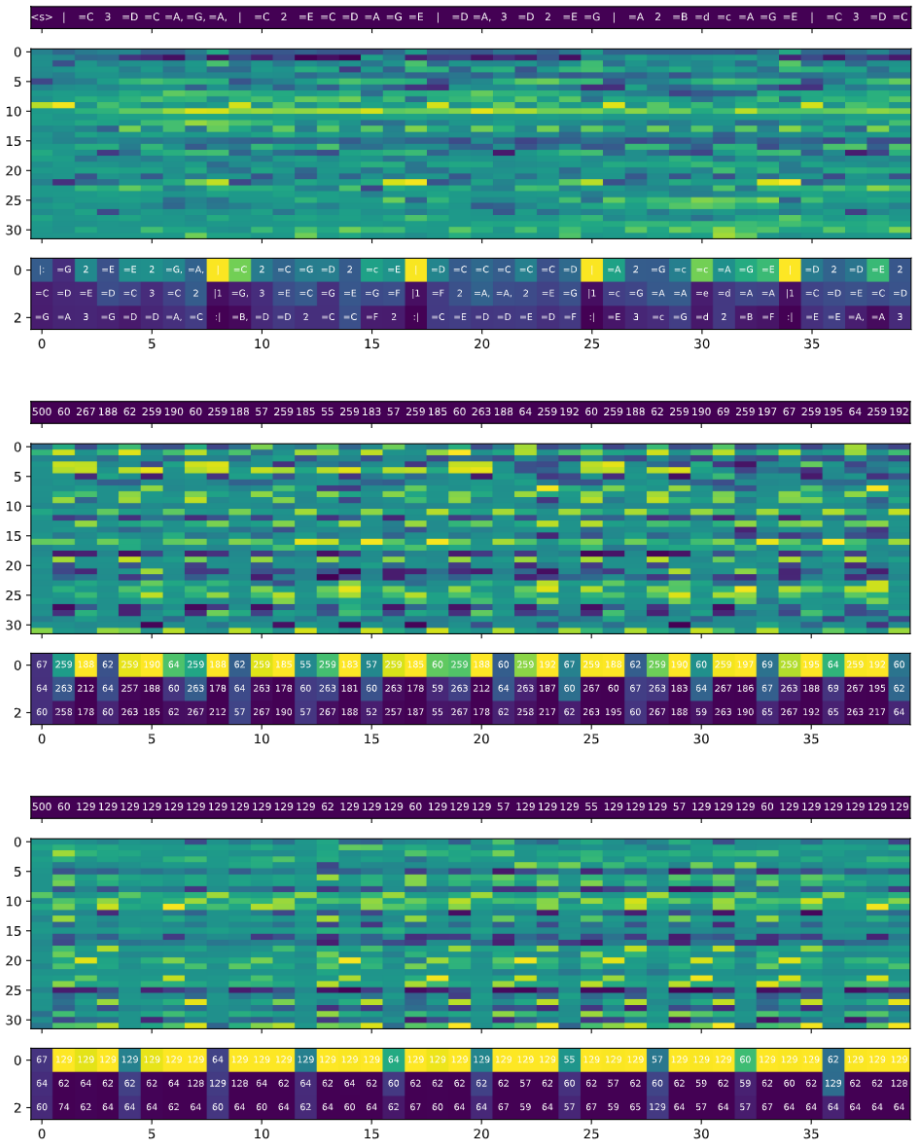


Fig. 2: Activations and predictions for the same song (top to bottom: ABC-event1-tstep1) For each of the three, on top: input sequence, middle: LSTM activations for each neuron (Dark blue is -1, Yellow is +1), bottom: top three predictions (more yellow is higher probability).

References

Byrd, D., Boyle, R. D., Berggren, U., Bainbridge, D., et al. (1997). *Beyond midi - the handbook of musical codes.* MIT press.

Dannenberg, R. B. (1993). A brief survey of music representation issues, techniques, and systems.

Eck, D., & Schmidhuber, J. (2002, 02). Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In (Vol. 12, p. 747 - 756). doi: 10.1109/NNSP.2002.1030094

Gers, F. A., & Schmidhuber, J. (2000). Recurrent nets that time and count. In *Proceedings of the ieee-inns-enns international joint conference on neural networks. ijcnn 2000. neural computing: New challenges and perspectives for the new millennium* (Vol. 3, pp. 189–194).

Honing, H. (1993). Issues on the representation of time and structure in music. *Contemporary Music Review*, *9*(1-2), 221-238. doi: 10.1080/07494469300640461

Karpathy, A., Johnson, J., & Fei-Fei, L. (2015). Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Oore, S., Simon, I., Dieleman, S., Eck, D., & Simonyan, K. (2020). This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, *32*(4), 955–967.

Plitsis, M., Kritsis, K., Kaliakatsos-Papakostas, M., & Katsouros, V. (2020). Evaluation of different symbolic encodings for music generation with lstm networks. In *Proceedings of the 13th international workshop on machine learning and music (mml2020) at the european conference on machine learning and principles and practice of knowledge discovery in databases (ecml/pkdd 2020)* (p. 41).

Sturm, B. (2018). What do these 5,599,881 parameters mean?: An analysis of a specific lstm music transcription model, starting with the 70,281 parameters of its softmax layer. In *International conference on computational creativity.*

Sturm, B. L., Santos, J. F., Ben-Tal, O., & Korshunova, I. (2016). Music transcription modelling and composition using deep learning. In *Proc. conf. computer simulation of musical creativity.* Huddersfield, UK.

Todd, P. M. (1989). A connectionist approach to algorithmic composition. *Computer Music Journal*, *13*(4), 27–43. Retrieved from `http://www.jstor.org/stable/3679551`

Yang, L.-C., & Lerch, A. (2018). On the evaluation of generative models in music. *Neural Computing and Applications*, 1-12.