Feature Evaluation of Music Generation using SampleRNN

Bart van der Braak and Albert Meroño Peñuela

Department of Computer Science, Vrije Universiteit Amsterdam, NL
bartvdbraak@gmail.com – https://bartvdbraak.github.io/
albert.merono@vu.nl – https://www.albertmeronyo.org/

Abstract. Deep Learning techniques, such as Neural Networks, have become of considerable significance to the domain of music. Especially for tasks as segmentation, recommendation and classification at first, but now also in terms of generation. To create in the non-symbolic music domain, we utilized SampleRNNs framework to generate raw audio. Multiple music datasets with varying complexity and timbre were trained upon. They are evaluated by comparing the feature set of the training dataset and of the respective generated results. A statistical test is used to determine that some features, such as Key and BPM, are fully preserved by SampleRNN.

Keywords: Recurrent Neural Network · SampleRNN · Music Generation · Feature Evaluation

1 Introduction

The non-symbolic music domain attracts attention from many artists, producers and musical platforms. The novelty of this domain being used in machine learning may also prove to be interesting for actual musicians that are interested in music generation through the use of Deep Learning algorithms.

The majority of developed musical machine learning models are based in the symbolic domain. While there are many advantages, the disadvantages of this domain, such as in MIDI, are the need for synthesizing, no vocals, limited audio effects and having less realistic sounding instruments. The use of non-symbolic music protocols with Deep Learning models may have equal or more interesting outcomes in their results and evaluation.

There has been work done in this domain by the team behind SampleRNN[4]. Their algorithms have been refactored into different frameworks by others, such as PyTorch[5]. The team behind Dadabots[2] has even created full-length albums of Black Metal and Math Rock using SampleRNN.

The SampleRNN team has shown the results of their testing and validation results in terms of loss functions. The Dadabots team has shown the artistic potential for musical forms that aren't as traditional. There hasn't been any evaluation done on the respective outputs in terms of musical features in comparison with their training datasets. Our research uses different musical features

to evaluate the extent to which SampleRNN preserves them. This research will use numerical and statistical measures instead of subjective measures to evaluate SampleRNNs quality of output.

This paper will address the following research question: "To what extent will generated music from SampleRNN preserve its musical features?".

To answer these questions, this paper will obtain the values for these features for the original datasets and their respective outputs.

The paper is organized in the following way. Section 2 references related work to our research. Section 3 describes the research method with its experiments. Section 4 covers the results of the different experiments. Section 5 handles our conclusion and the subsequent section 6 discusses our results as well as any notes on the method, difficulties and future work.

2 Related Work

2.1 Symbolic Domain

Symbolic music refers to note sequences. Their generation using machine learning is of lower computational complexity: training is a matter of minutes or hours.

MusicVAE[1] is a hierarchichal variational autoencoder that learns latent spaces for musical scores. It can be used to perform smooth and realistic transitions of music latent space interpolations.

MidiNet[7] uses Convolutional Neural Networks and turns it into a Generative Adverserial Network to train models that can generate MIDI music. Its performance was evaluated against Magentas MelodyRNN[2].

2.2 Non-symbolic Domain

Non-symbolic music is raw audio. Generally, it has a higher computational complexity as it takes hours or even weeks to gain significant results.

Google Deepminds WaveNet[6] uses Convolutional Neural Networks with raw audio as input to create raw audio. It is mostly used in speech applications but can also generate raw music audio.

NSynth[3] uses an autoencoder in the style of WaveNet[6] and a large musical notes dataset to learn multiple embeddings to generate different timbre.

2.3 Feature Evaluation

Essentia[1] is an open-source library written in C++ with Python bindings for audio analysis and music information retrieval. Its feature extractor is used for the evaluation in this paper.

[1] Magentas MusicVAE, `https://magenta.tensorflow.org/music-vae`

[2] Magentas Melody RNN, `https://github.com/tensorflow/magenta/tree/master/magenta/models/melody_rnn`

3 Research Method

This research paper outlines its method into four tasks: selecting the datasets used for training, setting up the resources and environments, training the models and evaluating the results.

3.1 Dataset Selection

For training the models, four different datasets were used: Two datasets are centered in the genre of Jazz and the other two in Techno. Within these two genres, the major difference between the datasets were the variety in musicians. For one, we used recordings of one artist and in the other there was a variety of artists recordings. Table 1 gives an overview of these datasets in terms of description, length and references to their content.

Table 1. Overview of the datasets that are used to train different models.

Dataset	Description	Length(min.)
Various Jazz Artists[3]	59 songs by various Jazz artists	337
Bunk Johnson[4]	96 recordings by 20th century Jazz trumpeter	308
Various Techno Artists[5]	47 songs by various Techno artists	315
Joseph Capriati[6]	One 6 hours long continuous Techno DJ-set	354

3.2 Setup & Resources

For running the experiments there was a requirement of specific hardware: a CUDA-capable GPU is recommended for running SampleRNN. We opted for two server instances with the configurations described in table 2.

The CUDA drivers are installed on the servers as well as SampleRNNs dependencies in virtual environments. SampleRNNs codebase is cloned into the server along with our datasets.

3.3 Training Method

First, the datasets are preprocessed using a custom script that concatenates all audio and then is split into 8 second audio chunks with a sample rate of 16kHz.

[3] Various Jazz Artists playlist, Spotify: `https://open.spotify.com/user/118412596/playlist/1lBY4hiI08LdFoIvvoixjn?si=TA9rMK5mTfCBhZqw-JpHzQ`

[4] Bunk Johnson playlist, Spotify: `https://open.spotify.com/user/118412596/playlist/2AF35OEN7RhOOsy96BgDOK?si=WhLB_V-eS4i8TJYNedSfTQ`

[5] Various Techno playlist, Spotify: `https://open.spotify.com/user/118412596/playlist/1LbLPtdiLA5CA9Rky86YpA?si=I2ABLqTYTtSGTu2I_koqog`

[6] Joseph Capriati at Spazio 900, Soundcloud: `https://soundcloud.com/joseph-capriati/joseph-capriati-spazio-900-roma-16-12-2017-6-hours-set`

Table 2. Overview of the server instances configuration.

RAM	7,5 GB
CPU	2 vCPUs, Intel Sandy Bridge
GPU	NVIDIA Tesla K80
DISK	50GB
OS	Ubuntu 16.04.4 LTS
ARCH	x86_64

The data is split into 88% training, 6% validation and 6% testing sets. The models configuration has the following defaults: a batch size of 128, 1024 dimensions, sequence length of 1024, 256 quantization levels, learn H0 and weight normalization.

After training, the result folders were retrieved from the server instances for evaluation.

3.4 Evaluation Method

For each of the datasets and their respective training and result data, we ran a custom script that runs `essentia_streaming_extractor_music` from Essentias extractor library over each audio file. This creates a folder with extractor data in JSON-formatted files.

Their algorithm computes a substantial set of spectral, time-domain, rhythm, tonal and high-level feature descriptors. For statistical analysis, the descriptors that were selected are `BPM`, `Danceability` and `Key`. These features are described in table 3.

Table 3. Descriptors from Essentias feature extractor that were used in the evaluation.

Feature	**Description**	**Class**
BPM	BPM value according to detected beats	Rhythm
Danceability	Estimation of Danceability derived from DFA	Rhythm
Key	Key estimate given a pitch class profile (EDMA)	Tonal

4 Results

Each of the experiments were trained for 2-4 days. The model improves upon by lowering its training loss and validation loss on every epoch and in between. The resulting samples are described in table 4, along with a link to their epoch samples.

Table 4. The resulting datasets with the amount of epoch samples generated and where to listen to them.

Dataset	Amount	Epoch Samples
Various Jazz	30	https://soundcloud.../dataset-various-jazz
Bunk Johnson	34	https://soundcloud.../dataset-bunk-johnson
Various Techno	13	https://soundcloud.../dataset-various-techno-1
Joseph Capriati	23	https://soundcloud.../dataset-joseph-capriati

4.1 BPM

In figure 1, we can see that BPM has different distribution shown in the boxplots. For the Jazz centered datasets, the boxplots whiskers are almost the same when comparing the result and training sets. Their median values have a difference of around 5-10 BPM.

For the Techno datasets, something different has happened. The training sets have a very narrow distribution for BPM, something that is to be expected for electronic music. However, the result sets appear to show a much larger variance, shown by their first and third quartiles. The median BPM of these sets are within a 5-10 BPM margin of each others respective value.

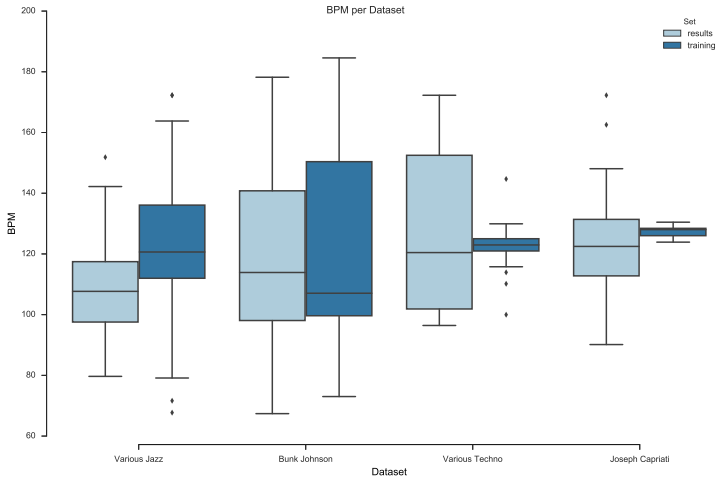

Fig. 1. This figure shows boxplots for the training and results sets for each dataset. BPM values are plotted on the y-axis and the different datasets on the x-axis.

4.2 Key

In figure 2, four frequency barplots are shown for each key value that was found in the datasets. These have substantial distribution differences between all the training sets and their respective result sets. All the results have a key value

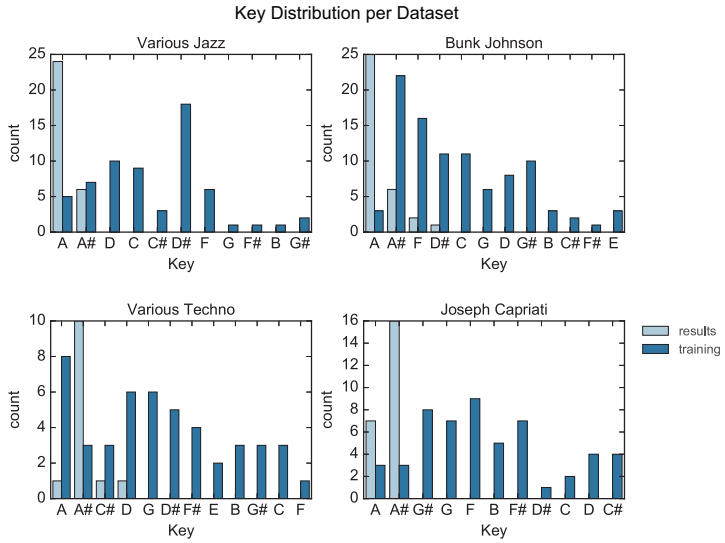

Fig. 2. This figure shows four barplots for the training and results sets for each dataset. Frequency (count) values are plotted on the y-axis and key values on the x-axis.

around A or A#. The training data, however, has a much more uniform distribution of key.

4.3 Danceability

Figure 3 shows boxplots of Danceability values within each of the datasets. Given the boxplots whiskers, the distribution of Danceability is almost the same for the Various Jazz and Joseph Capriati datasets. This is a bit less apparent with Bunk Johnson, with the lower tail differing 0.3. This might still be a reasonably small difference, given that these values generally reach values between 0 and 3. The Various Techno dataset doesn't seem to reflect Danceability values in the result set relating to the training set.

There were two extreme outlier values (4, 9) in the Various Techno (training) and Joseph Capriati (results) datasets. They weren't drawn into the plot for visibility reasons.

4.4 Epoch Progression

The progression between each epoch in terms of Danceability values is plotted in Figure 4. It is seems that, considering the training value, in the Bunk Johnson dataset this progression converges more correctly in the later epochs. The Joseph Capriati set seems to start and end with the same linear trend, close to the training value.

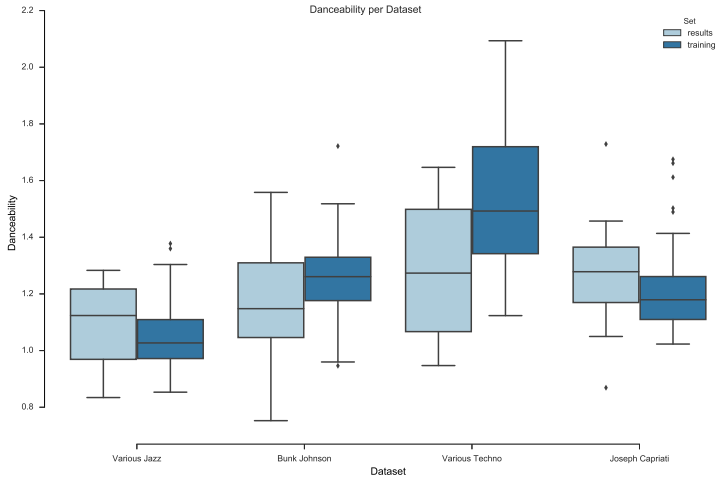

Fig. 3. This figure shows boxplots for the training and results sets for each dataset. Danceability values are plotted on the y-axis and the different datasets on the x-axis.

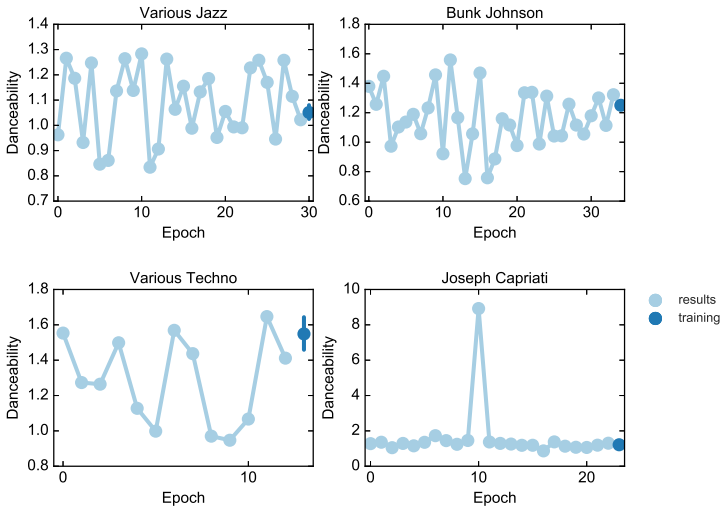

Fig. 4. This figure shows the Epochs progression of Danceability for the training and results sets for each dataset. Danceability values are plotted on the y-axis and the Epoch values on the x-axis. Training set values are shown within a singular x-value.

5 Conclusion

The results from our evaluation method lead to the conclusion that some features were more preserved than others. Also, there were some disparities between our findings of each of the datasets. We conducted a t-test on all our data to measure any significant differences between our *training* and *results* datasets. Our null-hypothesis is $\mu_t = \mu_r$ where μ_t is the mean of the training set and μ_r the mean of the result set. We use a significance level of 5%. The resulting p-values are shown in table 5.

Table 5. Our t-tests null-hypothesis, significance level and the p-values from each dataset and respective features. P-values that were smaller than our significance level are emphasized.

$H_0 : \mu_t = \mu_r$		*p-values*		
$\alpha = 0.05$		BPM	Key	Danceability
Datasets	Various Jazz	**0.0047**	0.2109	0.1808
	Bunk Johnson	0.8525	0.1558	**0.0035**
	Various Techno	0.4114	0.9311	**0.0097**
	Joseph Capriati	0.2351	0.5402	0.0937

For the Various Jazz dataset, BPM had a p-value that was lower than our significance level. Only in this case, we reject the null-hypothesis for BPM. There were no significant p-values for the other three datasets.

Key as a feature shows significantly different distributions. Where the training sets have more uniform distributions of key, the results were always narrowly distributed around the A and A# keys. However, we did not find any p-values under our significance level. Therefore, we cannnot reject our null-hypothesis in these cases.

In terms of Danceability, The Bunk Johnson and Various Techno dataset show two p-values that are lower than our significance level. Their null-hypotheses are rejected. The two other datasets were not significantly different.

Ultimately, our findings are that Key was fully preserved, BPM mostly preserved, while Danceability was preserved in half of the experiments. Therefore, SampleRNN has preserved all the features we used to measure, except in some specific cases.

6 Discussion

Our research has some pitfalls in terms of results. First of all, we are working with a relatively small set of epochs. This resulted in a large portion being very noisy audio. This can also be atrributed to the fact that the search for good initial hyperparameters takes alot of time. Especially when you are trying to run four different experiments in parallel.

Secondly, finding objective audio features is a whole discussion in and of itself. Audio based features are measured based on different algorithms. Where high-level ones can prove interesting findings, such as Spotify's Get Audio Features[7] (formerly The Echo Nest), these are mostly based on trained models. Their API is unaccessible when it comes to uploading audio files.

6.1 Legality

An interesting question to raise about these algorithms in conjunction with intellectual property: Is generated music through SampleRNN with the (exclusive) use of copyrighted material an infringement on said material?

The answer of this question is not completely straightforward. First of all, it depends greatly on how the output ultimately sound. Secondly, it depends on the intent of the created material. With commercial intent there would be a greater risk of claims than with research-based intentions.

6.2 Replicability

Some resources will be available for those that are interested in replicating this research in any sort of way. In terms of running the algorithms for SampleRNN, a blog post was created with detailed instructions on how to set up a (server) environment that can run SampleRNN[8].

For running our custom scripts, we have a respository with all the code[9]. These include scripts and a guide for data preprocessing and feature extraction.

If you are in need of hardware/server resources, I would suggest looking at cloud servers that allow you to run code on machines with GPU support. At this point in time, Google Cloud Service provides a free 12-month 300$ credit[10].

References

1. Bogdanov, D., Wack, N., Gómez Gutiérrez, E., Gulati, S., Herrera Boyer, P., Mayor, O., Roma Trepat, G., Salamon, J., Zapata González, J.R., Serra, X.: Essentia: An audio analysis library for music information retrieval. In: Britto A, Gouyon F, Dixon S, editors. 14th Conference of the International Society for Music Information Retrieval (ISMIR); 2013 Nov 4-8; Curitiba, Brazil.[place unknown]: ISMIR; 2013. p. 493-8. International Society for Music Information Retrieval (ISMIR) (2013)
2. CJ Carr, Z.Z.: Generating black metal and math rock: Beyond bach, beethoven, and beatles (2018)

[7] `https://developer.spotify.com/documentation/web-api/reference/tracks/get-several-audio-features/`

[8] Setting up a VM for Machine Learning `https://bartvdbraak.github.io/posts/setting-up-vm-for-machine-learning`

[9] `https://github.com/bartvdbraak/FeatEvalMusGen`

[10] `https://cloud.google.com/free/`

3. Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., Norouzi, M.: Neural audio synthesis of musical notes with wavenet autoencoders. arXiv preprint arXiv:1704.01279 (2017)
4. Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A., Bengio, Y.: Samplernn: An unconditional end-to-end neural audio generation model. arXiv preprint arXiv:1612.07837 (2016)
5. Piotr Kozakowski, B.M.: Samplernn in pytorch. `http://deepsound.io/samplernn_pytorch.html`
6. Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499 (2016)
7. Yang, L.C., Chou, S.Y., Yang, Y.H.: Midinet: A convolutional generative adversarial network for symbolic-domain music generation. In: Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR2017), Suzhou, China (2017)