Language Modelling for Concatenative Synthesis

Creative audio generation tool

Computing Final Project Preliminary Report

Wojciech Kacper Werkowicz
Student ref. 33655491

> *Early draft of an epigram*
> Every word begins
> In the Empty Words
> And ends in it
> ///
> Every word begins
> And ends
> In itself
> ///
> The Empty Word
> Begins or ends
> Only in itself
> ///
> A.A. Markov, *Theory of algorithms* (translation: 1961), p.20
> - C.C. Hennix, *Poësy Matters*

Introduction

This project is an attempt of implementing a software tool capable of generating creative and coherent variations/extensions of analysed audio signal using concatenative synthesis, enhanced by generative language

modelling.
Its motivation is to investigate the combination of methods which, while not covered by recent audio related deep learning projects, hints at the possibility of generating high quality output without much computational power required, greatly improving the accessibility of recent deep learning developments for creative sound practice.
Concatenative synthesis in its basic form is completely context-agnostic, therefore what is being sought in this project is a method for generating context-aware trajectories across the space of sound corpora.
This is a task suitable for neural network architecture capable of sequence generation via predicting the next possible token based on previous ones, which is called a *language model*. Intuitively, this would mainly suggest its ability to process text, though it can be efficiently generalised to model any kind of sequence data, sound and music not being exceptions[1].
The fundamental idea of this project is to granulate a given audio signal, learn its statistical *latent space* and generate a sequence of arbitrary length based on that model.

Background Research

Concatenative synthesis

A given input signal can be easily split at transients or regular intervals into chunks called *grains* which can then be played back, in a method described as granular re-synthesis or *granulation*[2]. An extended approach to it undertakes an analysis of each grain content, extracting multiple high level features (also called descriptors) values of which can then serve as N-dimensional coordinates, organising the chunks of what was initially an audio sequence into a multi-dimensional space open for creative exploration[3].
Concatenative audio synthesis has a long history of utilising ML methods to creative sound practices, as they are crucial for tasks such as pre-processing, extraction of more complex features, interpretation large amounts of descriptor data as well as dimensionality reduction, often necessary for the ease of human interaction. Hence Corpus-Based

Concatenative Synthesis toolboxes such as IRCAM's CataRT[4] and FluCoMa[5] developed at University of Huddersfield naturally provide advanced tools for audio decomposition, analysis and re-synthesis. In case of CataRT, various approaches to predicting sequences comprehensively have been attempted using methods such as hidden Markov models as well as non-ML based weighted context-aware methods working in real-time[6], while FluCoMa's wide availability between different environments (Max MSP, PureData, SuperCollider or even command line) extends itself easily with any affordances available within given software.
The strong sides of classic concatenative approach are its convenient interface (a 2D plot is commonly used to display the corpus space after applying dimensionality reduction algorithms), as well as the efficiency of its use in real time: the source audio file is loaded only once and it is segmented only symbolically using start and end indices for each slice. The grains can then be polyphonically played back from the same buffer as the original file. Notably, some concatenative synthesis approaches focus on live input use.
For this project, a simple form of concatenative synthesis proves not to be enough - as mentioned in the previous section, it is almost completely agnostic to the context of grains by default and is incapable sequence generation on its own.

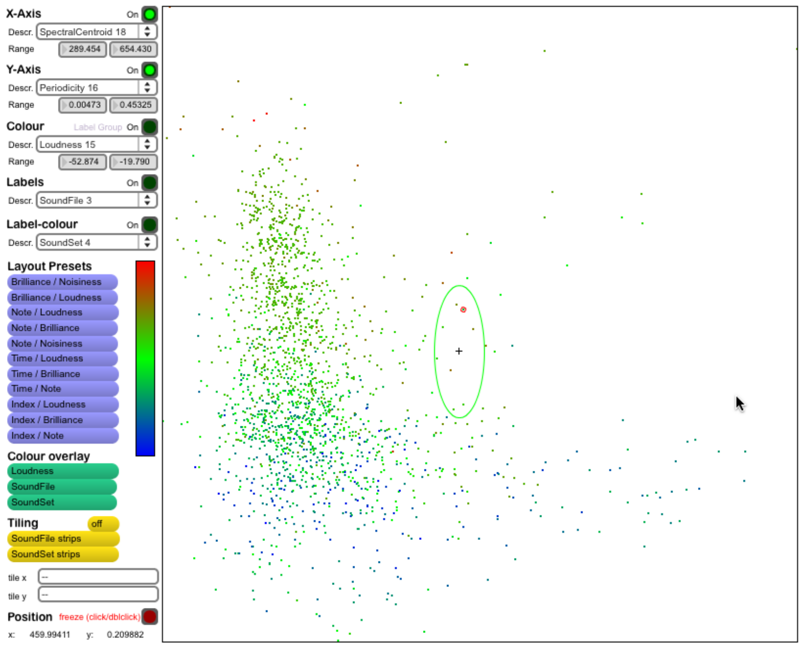

Fig. 1 - *2D display of catart.lcd5*

Variational Auto-Encoders

A decade later, with increasing popularity of the Deep Learning branch of Machine Learning research, a successful attempt has been made to replace the calculated descriptors of audio grains with learned latent representations using variational auto-encoders (VAEs). Neural Granular Sound Synthesis[7] which was developed at IRCAM, had been followed in 2021 by a a real-time implementation of an audio VAE - RAVE[8] which also uses a Generative Adversarial Network (GAN) to model trajectories between latent representations of signal frames, enabling non-conditioned generation. While the signal generation runs efficiently (and, importantly, faster than real-time) on an average consumer CPU, training a generative model with RAVE still takes several days on a high-end GPU and requires plenty of training data (>=3h). More so, it actually consists of two models which need to be fitted consequently - the first responsible for encoding and decoding latent representations, second for modelling trajectories across the former model's latent space.

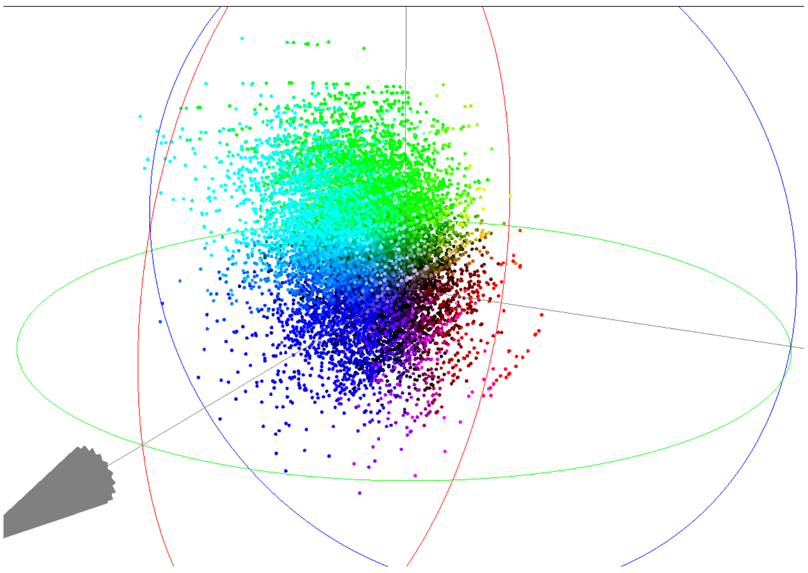

Fig. 2 - *3D plot of a latent space of a custom generative (prior) RAVE model trained on ~40h of drum break recordings (first three latent dimensions are mapped to XYZ and RGB values of plotted points)*

Transformer-based Language Modelling

Two years before, OpenAI released MuseNet[9] and then JukeBox[10] transformer-based networks, which apply generative language modelling to sequences of MIDI notes extracted from fed samples. In case of the latter, multiple characteristics can be extracted and injected, such as arrangement, style, lyrics, etc. Generated sequences of MIDI notes are then fed into WaveNet[11], which calculates the output raw audio on per-sample basis. While transformer-based architectures are considered to be state of art at the moment (most recent example being the wildly popular ChatGPT[12]), Being this complex, that this kind of pipeline proves very computationally intensive during inference is no surprise, while resources for fitting a model of such large scale are beyond the reach of an individual. Fine-tuning one of many models provided by OpenAI seems the most accessible option available - these span across work of various artists and genres but this, again, requires a suitably powerful GPU runtime. At the same time these still no doubt include certain assumptions about what music is, which come from the choice of training data and process of feature engineering.

Fig. 3 -"PLAYSTATION STARTUP," Continued by a Neural Network (OpenAI's Jukebox))

Research Plan

As described above, common approach in applying Deep Learning [DL] to audio generation relies on symbolic representations of sound such as MIDI, which assumes a very specific way of engaging with the source material - that is as something possible to transcribe, but also standardise into a protocol which inherently contains notions of what music/sound can be.

On the other hand, while some of the methods for neurally generating raw signal come out of CataRT, they mostly requires significant computing resources, which are not easily accessible and is difficult to experiment with for an individual with no expertise in DL.

Method

A solution to this would be to combine these two approaches and use CataRT style feature extraction, then training a possibly small neural network on the trajectories across the sound corpus present in input material in its original, sequential form.
The suitable architecture for this task seems to be a generative Language Model[13], based either on Long-Short Term Memory [LSTM] or a Transformer network.
Modelling the sound corpus statistically, it should be possible to sample from its prediction space, generating trajectories which are coherent and alike to the organisation of the grains in the original audio.
Predicted sequences (trajectories) can be approached similarly as in CataRT - omitting synthesis of raw audio, as the original data keeps on being stored in the memory and short chunks of it can be recovered with ease. These can be then played back in the manner of granular re-synthesis, with a set of user adjustable parameters.
As mentioned, this type of model architecture should be small and light, with training and inference within the reach of a consumer-grade laptop CPU and a trained model could be wrapped in a Max MSP external object, a VST plug-in using TorchScript[14] or an equivalent, in cases of using TensorFlow for designing the architecture and training.
This should enable inference performant enough for real-time use without much Python familiarity (except for, perhaps, running a wrapper script on a trained model), where the model is only responsible for generating trajectories across the corpora with parametrised temperature (diversity) and the rest of the tasks, such as sample playback etc. can be implemented by the user to their liking. This can be compared to MuBu[15] and FluCoMa Max packages which function as toolkits which can be appropriated into existing workflows rather than provide complete environments.

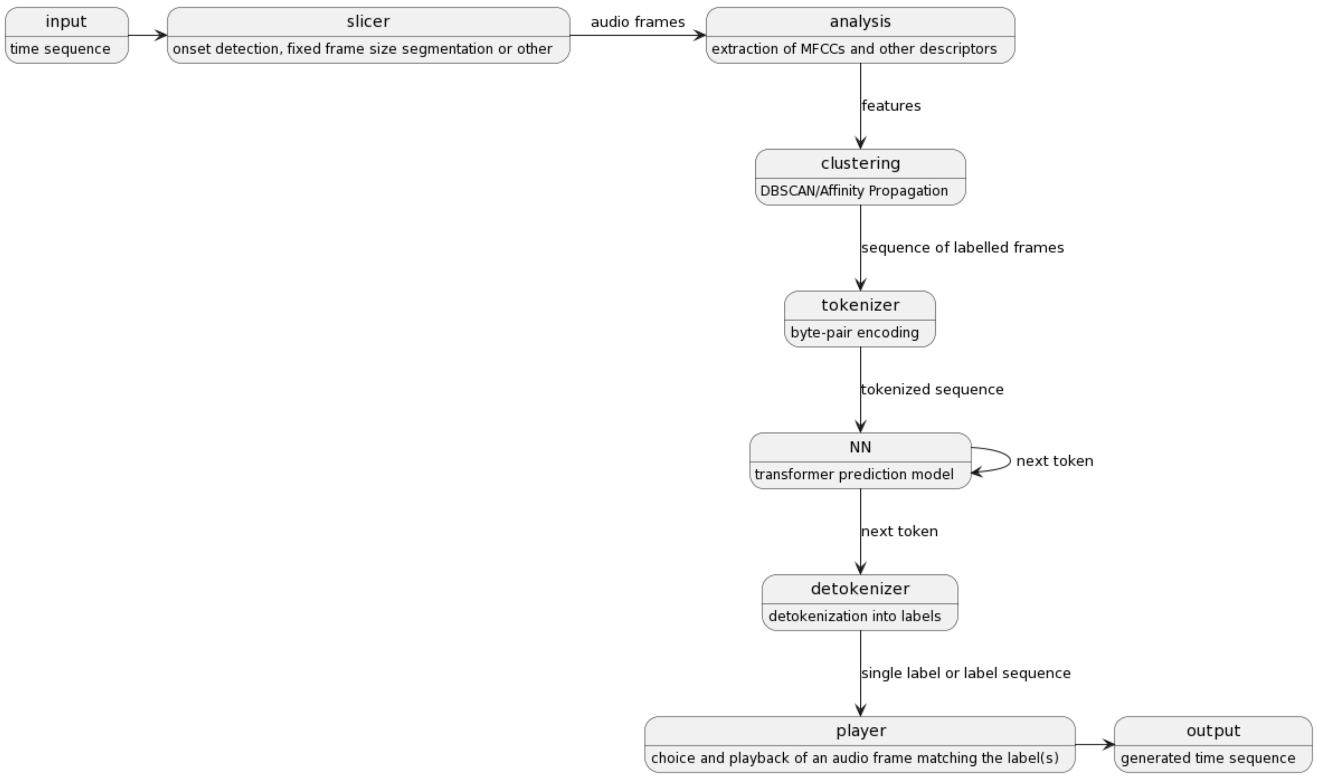
input
time sequence
slicer
onset detection, fixed frame size segmentation or other
audio frames
analysis
extraction of MFCCs and other descriptors
features
clustering
DBSCAN/Affinity Propagation
sequence of labelled frames
tokenizer
byte-pair encoding
tokenized sequence
NN
transformer prediction model
next token
next token
detokenizer
detokenization into labels
single label or label sequence
player
choice and playback of an audio frame matching the label(s)
output
generated time sequence

Fig 4. - *prototype data flow diagram*

Proof-of-Concept

Implementation

Appendix 1. is a Jupyter notebook presented as a naive implementation of the described project. Given roughly 4 minutes of a drum track separated from Lyn Collins' *Think (About It)*[16] audio segmentation and extraction of crude features (energy and zero-crossing rate) are executed using librosa[17]. An N-th order Markov chain serves in the place of a generative model. Results are then re-synthesised as a stream of grains emitted at regular intervals, similarly to Curtis Roads'[18] synchronous granular synthesis method.

Evaluation

Perceptually, there is plenty of structural coherence to the generated sequences, though a large part of the quality of the output depends on the way re-synthesis is approached and the manner in which it impacts the rhythm. A big issue here is the lack of a viable method for benchmarking, as a *coherent sequence* proves to be a very subjective notion as it heavily depends on scale: a coherent arrangement of grains

within a cloud relates more to what human ears perceive as timbre and differs to a case of generating beats (ie. drum hits) within a bar.
This raises more questions than answers, such as:
What should be target material? Is the quality sought after one possessed by acoustic sound sources? Is it timbral or rhythmic?
Answering these prior questions is nothing but urgent for advancing this project.

Evaluation Plan

Evaluation Questions

- Can an arbitrary sound corpus be meaningfully segmented, analysed and clustered using standard Music Information Retrieval (MIR) tools?
- Can causal relationships between elements of said corpus series be efficiently modelled and predicted using a Language Model?
- Given a prediction of such a model, is it possible to synthesise an audio sequence using the original corpus with organic sounding results?
- Can such a model be successfully used for real-time ?
- Is it possible to develop a general methodology for achieving the above?
- If any of the above are successful, what are pre-requisites for satisfying results in terms of type and amount of data provided, as well as scale on which the analysis is provided?

Evaluation Methods

- Empirical judgement of coherence of clustered corpus elements
- Blind comparison tests between human-made and generated sequences
- Audio-specific accuracy metrics for evaluating the re-synthesis method such as ones related to spectral energy[19]
- Performance benchmark of real-time inference
- Testing against a broad range of dataset sizes and datasets with varied character (ie. speech, electronic music, instrumental

recordings, field recordings, one shot sounds)
- Testing against a broad range of scales, from micro-sound to entire sections of musical pieces

Development Schedule

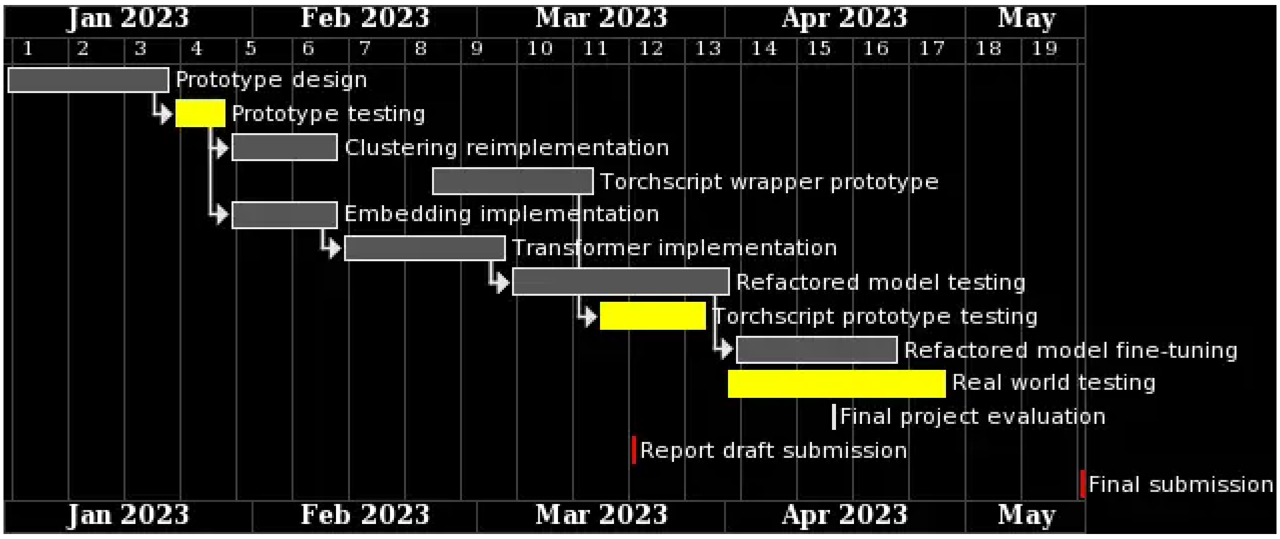

Fig 5. - *timeline of project development schedule*

Appendices

Appendix1.html
HTML export of a Jupyter notebook implementing a toy generation tool as a proof-of-concept.

---

1. Chollet, François. *Deep Learning with Python*. Second edition. Shelter Island: Manning Publications, 2021, p. 365-366.↩
2. Roads, Curtis. *Microsound*. 1. paperback ed. Cambridge, Mass.: MIT Press, 2004, p. 187-193.↩
3. Schwarz, D. 'Corpus-Based Concatenative Synthesis'. *IEEE Signal Processing Magazine* 24, no. 2 (March 2007): 92–104. https://doi.org/10.1109/MSP.2007.323274.↩
4. Schwarz, Diemo. 'CataRT'. Ircam. Accessed 30 January 2023. http://catart.concatenative.net/.↩
5. Tremblay, Pierre Alexandre, Gerard Roma, and Owen Green. 'Enabling Programmatic Data Mining as Musicking: The Fluid Corpus Manipulation Toolkit'. *Computer Music Journal* 45, no. 2 (1 June 2021): 9–23. https://doi.org/10.1162/comj_a_00600.↩

6. Schwarz, Diemo, Sylvain Cadars, and Norbert Schnell. 'What Next? Continuation in Real-Time Corpus-Based Concatenative Synthesis.' In *International Computer Music Conference (ICMC)*, 1–1. Belfast, United Kingdom, 2008. https://hal.archives-ouvertes.fr/hal-01161402.↩
7. Bitton, Adrien, Philippe Esling, and Tatsuya Harada. 'Neural Granular Sound Synthesis.' arXiv, 3 July 2021. http://arxiv.org/abs/2008.01393.↩
8. Caillon, Antoine, and Philippe Esling. 'RAVE: A Variational Autoencoder for Fast and High-Quality Neural Audio Synthesis.' arXiv, 15 December 2021. http://arxiv.org/abs/2111.05011.↩
9. McLeavey Payne, Christine. 'MuseNet.' *OpenAI Blog* (blog), n.d. https://openai.com/blog/musenet/.↩
10. Dhariwal, Prafulla, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. 'Jukebox: A Generative Model for Music.' arXiv, 30 April 2020. http://arxiv.org/abs/2005.00341.↩
11. Oord, Aaron van den, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 'WaveNet: A Generative Model for Raw Audio.' arXiv, 19 September 2016. http://arxiv.org/abs/1609.03499.↩
12. OpenAI. 'ChatGPT: Optimizing Language Models for Dialogue.' Accessed 16 January 2023. https://openai.com/blog/chatgpt/.↩
13. Chollet, François. *Deep Learning with Python*. Second edition. Shelter Island: Manning Publications, 2021, p. 364-367.↩
14. Caillon, Antoine. 'RAVE.Js.' Accessed 16 January 2023. https://caillonantoine.github.io/ravejs/.↩
15. 'MuBu.' Ircam. Accessed 22 January 2023. https://forum.ircam.fr/projects/detail/mubu/.↩
16. Collins, Lyn. *Think (About It)*. LP. Monarch Pressing, 1972. https://www.discogs.com/master/52645-Lyn-Collins-Think-About-It.↩
17. McFee, Brian, Alexandros Metsai, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Frank Zalkow, et al. 'Librosa/Librosa: 0.9.2.' Zenodo, 27 June 2022. https://doi.org/10.5281/ZENODO.6759664.↩
18. Roads, Curtis. *Microsound*. 1. paperback ed. Cambridge, Mass.: MIT Press, 2004, p. 93.↩

19. Steinmetz, Christian J., and Joshua D. Reiss. 'Auraloss: Audio-Focused Loss Functions in PyTorch'. Centre for Digital Music, Queen Mary University of London, 15 December 2020. [auraloss: Audio-focused loss functions in PyTorch](https://doi.org/auraloss: Audio-focused loss functions in PyTorch).↩