AudioStellar, an open source corpus-based musical instrument for latent sound structure discovery and sonic experimentation

Leandro Garber
UNTREF
lgarber@untref.edu.ar

Tomás Ciccola
UNTREF
tciccola@untref.edu.ar

Juan Cruz Amusategui
UNTREF
jcamusategui@untref.edu.ar

ABSTRACT

Generating a visual representation of short audio clips' similarities is not only useful for organizing and exploring an audio sample library but it also opens up a new range of possibilities for sonic experimentation. We present AudioStellar, an open source software that enables creative practitioners to create AI generated 2D visualizations of their own audio corpus without programming or machine learning knowledge. Sound artists can play their input corpus by interacting with learned latent space using a user interface that provides built-in modes to experiment with. AudioStellar can interact with other software by MIDI syncing, sequencing, adding audio effects, and more. Creating novel forms of interaction is encouraged through OSC communication or writing custom C++ code using provided framework. AudioStellar has also proved useful as an educational strategy in courses and workshops for teaching concepts of programming, digital audio, machine learning and networks to young students in the digital art field.

1. INTRODUCTION

Low dimensional representations of a high-dimensional corpus of data allows us to discover latent relations (i.e. patterns) that are difficult or impossible to grasp with other methods [1, 2]. Using machine learning (ML) for accomplishing this task is a common practice nowadays and it is done with all kinds of high dimensional data.

Short audio clips can also be represented this way resulting in a 2D point map where similar audio clips are clustered together (Figure 1).

We found that this representation is not only useful for organizing and exploring an audio sample library but when combining it with an user interface can also be used as a musical instrument. The concept of a 2D space (i.e latent space) where sounds lie is an interesting abstraction that redefines the artist relation with the sonic material and a fertile terrain for research. An intuition on this topic can be found in section 2.1.

We are studying the possibilities for playing and composing music that arises from thinking in this direction hence we created a software that is geared around this idea. AudioStellar is free and open source, available for Linux, Mac and Windows and it allows users without ML or programming knowledge to experiment with these concepts. We provide three initial modes that lets the user interact with latent space in different ways. In section 2 we describe the software and expand on this idea.

While ML is becoming part of everyday life, fundamental concepts and concerns are still out of reach for the general public. AudioStellar aims to introduce some of these topics while proposing new ideas related to sonic experimentation for creative practitioners. We are including the use of this software as an educational strategy in courses, talks and workshops for teaching initial concepts of programming, digital audio processing, machine learning and computer networks to young students and enthusiasts in the digital art field. AudioStellar can also be used in engaging experiences in the context of jams and hackatons.

Accordingly, AudioStellar is free, open source and completely hackable, meaning that studying, modifying, distributing and building upon is not only encouraged but an essential part of this project.

This paper is organized as follows: Section 2 describes the software capabilities, Section 3 offers a short insight on our experiences with different sound corpus, Section 4 is about our experiences using the software in different educational contexts and finally in Section 5 we discuss future work.

1.1 Related software

Various software has been developed around audio similarity and corpus-based sound creation. Early examples include soundspotting [3] systems like *Caterpillar* (2000), *Musaicing* (2001) and *SoundSpotter* (2004).

Unlike AudioStellar, these programs were mainly focused on experimenting different techniques for concatenative synthesis.

CataRT (2006) introduced an interactive 2D visualization which allowed to browse and play audio impulses using two user-selectable descriptors[4]. In our software and in the same way as *FluidCorpusMap* [5], the user doesn't have to select which two descriptors work best for the selected dataset since X and Y axis are data-driven and learned by the dimensionality reduction algorithm (more on this in 2.5).

Recent software like *Infinite Drum Machine* [6], *klustr* [7] and *XO* [8] make use of dimensionality reduction techniques like t-SNE [1] to render a 2D map where similar sounding audio clips are placed near each other. The first one is a web experiment, the second one is for exploration only and doesn't feature novel ways of interacting with the

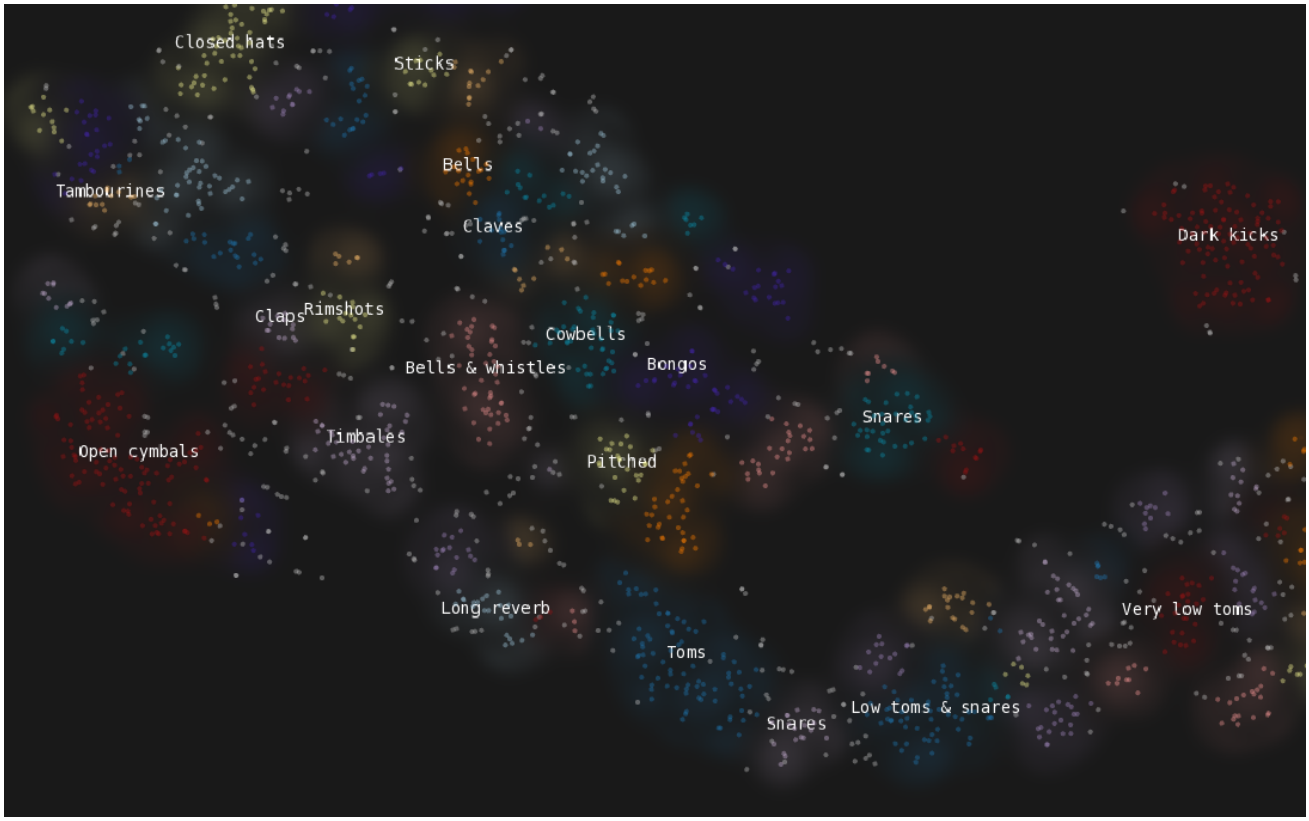

Figure 1. Screenshot. Similar audios are clustered together in latent space. Some human labeled groups found in default dataset.

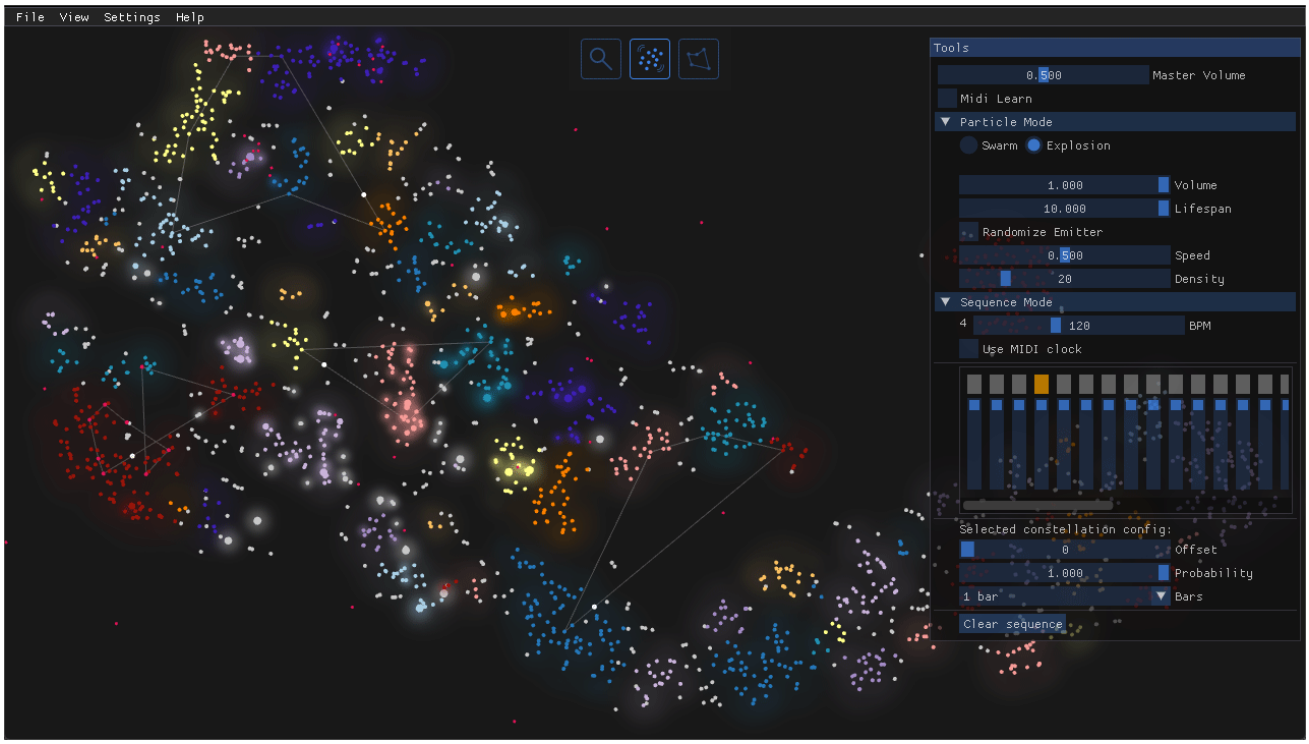

Figure 2. Screenshot. Using different modes to interact with latent space.

visualization and the third one is a commercial software not available for Linux centered around rythmic patterns creation.

Wekinator [9] is about supervised learning and live performance. Users without programming or ML skills can train models that output a signal for controlling any OSC compatible system (e.g. synths, DAWs or custom applications). While *Wekinator* is being used for sonic experimentation [10, 11], it is also an interesting tool for ML education [12]. This twofold aim inspires us to develop our unsupervised, sound specific approach.

2. SOFTWARE DESCRIPTION

We have put together an unsupervised learning pipeline to generate an interactive 2D point map that allows to browse and play audio samples in novel ways using various innovative sonic exploration modes.

The software requires no programming or ML skills and can process a user-selectable folder containing audio files to generate a *sound map* placing each audio file as a point in a 2D space. Nearby points correspond to spectrally similar sounds while far away points are dissimilar ones. This enables the creative practitioner to explore a sound library or field recording aided by AI that reveals a latent structure present in the input sound files.

Current version is 0.10.0.

It is free, open source, cross-platform and can be downloaded at `http://audiostellar.xyz`

Source code can be found at `http://gitlab.com/ayrsd/audiostellar`.

2.1 Latent space

The space learned by the ML pipeline (i.e where 2D points representing the audio clips lie) is called latent space. It is a representation of the input audio clips that are encoded for achieving a semantically (e.g dark, resonant, noisy) meaningful space. Computer suggested semantics are data-driven and will adapt for each set of audios. X and Y axis are usually hard to interpret but audios with similar timbre, pitch, amplitude, envelope or a mix of these will cluster together forming groups that the user can interpret. As can be seen, complex sound characteristics that are relevant to each dataset are learned, unveiling a structure of relations between the sounds that was already present in the data itself but hard to grasp.

2.2 Sampling as a journey

Visual artists using other ML systems (e.g. GAN architectures [13]) are discussing heuristics on how to traverse latent space in a meaningful, creative way [14, 15]. Traversing latent space in this context means generating sequences of images from an input trajectory in an infinite often high-dimensional space.

Although in our context only points generated from the input corpus are playable (see 5) similar considerations apply to AudioStellar as multiple trajectories can also be defined. Playing samples becomes a journey through latent space as the user experiments with included modes for traversing it. Custom heuristics are possible through OSC or coding a custom mode in C++ (see 2.4.2).

2.3 Modes

AudioStellar features three modes, an exploration mode for analysing the generated map and listening to the learned relations, a particle mode that lets the user cast configurable moving particles that play the samples as they move in 2D space resembling granular synthesis, and finally a sequence mode that lets the user draw constellations made from the points on the map and play them in a sequenced fashion using distance as rhythm. Figure 2 is a screenshot of these modes operating all at once.

2.3.1 Explorer mode

This is the default mode. Dragging the mouse through the screen will play sounds and explore the sound corpus. In this mode, sounds can be mapped to MIDI notes to be played by a MIDI controller or software. It also supports touchscreens that allows an expressive way to interact with the software using the fingers.

2.3.2 Particle mode

This mode allows to cast particles that traverse space and play nearby sounds (when the distance between a particle and a sound is below a threshold it will play). Its basic parameters are: the lifetime (how much time until particles disappear) and an individual volume (independent from the master volume).

This mode offers different *models* to choose from, which change how particles move in space:

- Swarm model: particles display random walk-style erratic behaviour. Additional parameters include the magnitude of the resulting *jiggle* and its velocity (direction and speed of movement).
- Explosion model: particles are cast in an explosion-like radial expansion. Additional parameters include the density (number of particles that are cast with each emission) and speed in which particles radiate from the center point.

Using the MIDI protocol, one can assign a region of the space to a MIDI note and thus, cast particles using a MIDI controller. This, paired up with the possibility of assigning mode parameters to MIDI Control Change (CC) messages, opens up different possibilities for live performing.

This mode is specially suitable to create sound textures and clouds, specially when setting particle groups to move slowly through space, thus generating slowly evolving sounds due to timbrical similarities in nearby points.

2.3.3 Sequence mode

This mode allows to create *space aware* rhythmic sequences. Using the mouse cursor you can select sounds and form a constellation. This will be interpreted by the software as a rhythmic sequence where the timing is defined as a quantization of the distance between the sounds forming it. You can have multiple sequences running in parallel. Each sequence have multiple parameters to control: Volume, Bars (defining overall sequence length relative to a global clock in Beats Per Minute), offset (offsets the sequence forward by a sixteenth note), probability (of the next sound being played or not). This set of parameters can all be controlled via MIDI.

2.4 Interaction with hardware and software

2.4.1 MIDI

Users can plug their hardware controllers or connect with other software using the MIDI protocol. Tempo syncing is also available via MIDI clock.

Sounds can be mapped to a MIDI note and faders to CC messages using built-in MIDI Learn feature. *Particle Mode* allows to map MIDI notes to regions in space where particles will be emitted (see Section 2.3.2).

2.4.2 OSC

Mobile devices can be connected as surface controllers using popular apps like TouchOSC. Nearly all faders can be easily controlled this way. We provide a TouchOSC template to get started.

Another interesting range of possibilities comes from taking advantage of other existing programming languages to control the software through its OSC API. This allows any user to develop new and interesting ways of traversing the latent space and controlling existing parameters. We provide PureData, Max, Python and Processing examples to get started but any language that has an implementation of OSC can be used. We have seen artists using this feature for creating their own physical musical interface, sound object but also installations, generative works and performances.

Another good example that has an early support in the last version of the software is the live coding environment TidalCycles [1] . Since the language itself and its sound engine (a custom made library programmed in SuperCollider) are totally decoupled, it was possible to replace it with AudioStellar. This early implementation takes advantage of the possibility of naming clusters and then, from TidalCycles, treat them as sample folders. The idea is to be able to control other parameters of sounds in the future (gain, speed of playback, pan, effects, etc).

2.4.3 Routing audio output

Linux users can use JACK audio connection kit to connect its audio output to any DAW or effect rack to record and add effects. Similarly, Mac users can use SoundFlower and for Windows, Voicemeeter . It is possible to route individual sounds or clusters to specific outputs so as to assign different effects to each.

2.4.4 Ableton Link

We are considering adding Ableton Link support for syncing with other popular software. In the meantime, it is possible to MIDI sync AudioStellar with other software that already support Ableton Link.

2.5 Machine learning pipeline

The user selects a folder and the program will search for MP3 or WAV files in that folder and any subfolder. The audio data from each file is truncated to a user selectable length (or zero-padded if it is shorter). For stereo files the right channel is discarded. Feature extraction stage is executed for each audio file and the resulting matrix is vectorized, converted to a long vector. User can choose any combination between Short-term Fourier Transformation, Mel Frequency Cepstral Coecients, spectral centroid, chromagram and Root Mean Squared amplitude. Vectors are stacked together obtaining a matrix where rows represents audio files. The first dimensionality reduction stage is performed using principal component analysis (PCA). The program keeps as many components as needed for explaining 98% of the variance. The software then apply a second dimensionality reduction stage for obtaining just 2 variables for each audio file. Options are t-SNE[1], UMAP[2] or PCA. The result is then saved to a JSON file that will also contain user's session information.

Clustering is performed in 2-dimensional space and it is used for coloring the points. We are using DBScan [16] algorithm and it is possible to modify its parameters from the main application.

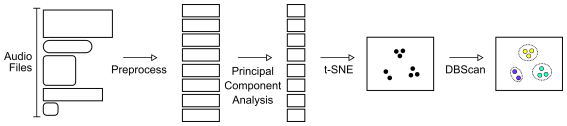

Figure 3. Machine learning pipeline.

Current pipeline runs well on any modern laptop and it doesn't require GPU. Refer to Figure 3 for a descriptive diagram.

2.6 Help

The software features a basic tutorial to get users started quickly and all buttons and faders have a corresponding help tooltip. Video tutorials will be available on YouTube.

We started a community on Facebook [2] to share experiences about AudioStellar, ask questions, keep in touch for new releases and discuss new features. Also, a mailing list [3] is available.

2.7 Programming specifications

We have chosen to build our application using C++ framework openFrameworks [4] to use a tool that is already familiar in the digital arts field. We are also using following addons: ofxAudioFile, ofxConvexHull, ofxImGui, ofxJSON, ofxMidi, ofxOsc, ofxPDSP, ofxSimpleTimer and ofxTweener.

The ML pipeline is written in Python using libraries sklearn [17], scipy [18] and librosa [19] among others. Everything is packed together using pyinstaller [20].

3. CORPUS BASED SONIC EXPERIMENTATION

The sonic result that a user can hear from the software will be in part defined by the heuristics used to traverse latent space but it is evident that the sound corpus being used will be the main factor.

The first approximation to test the software was using short percussive sounds from a drum machine sample pack. This is the default dataset that is shipped with AudioStellar and it gives a clear understanding of how the ML pipeline is performing. This kind of corpus is great for rhythmic

[1] `https://tidalcycles.org/`

[2] `https://www.facebook.com/groups/audiostellar/`

[3] audiostellar@googlegroups.com

[4] `http://openframeworks.cc/`

experimentation assisted by the computer but it puts out much of AudioStellar's possibilities.

As our understanding grew, we started experimenting with other types of corpus. Choosing a long recording and cutting it by a fixed length (or on its onsets) and then using the resulting bits as a new corpus is one of our favorites. Constellations of similar sounds unveil and it is possible to analyze any piece timbre and learn from it. Of course using this material for a remix is also great. For example chopping a singing track into short samples will result in a map that separates vowels, consonants, breathing, empty spaces and micro noise. One can then use sequence mode for a rythmic base or particle mode for getting a sonic cloud.

Other experiments include using a corpus of different acoustic instruments playing a scale and create musical phrases or pitched clouds, etc.

Much of this kind of experimentation is yet to be done and this is definitely part of our future research.

4. EDUCATION

Last year we included AudioStellar in courses, workshops and talks. One of our motivations for including AudioStellar in these is to introduce hard science topics and technical skills in a creative, ludic, hands-on manner.

We would like to encourage people from artistic fields to appropriate this new technology from a critical perspective for new ideas to emerge. Likewise, we are interested in people from hard sciences fields to use the tools they are currently using from a new perspective.

In this sense, our AI & Art course was a perfect place for students from both types of disciplines to meet and work together.

4.0.1 AI & Art course

It was an 8-days weekly crash course [5] that we gave twice last year where students learn the fundamentals of ML and we offer tools and ideas to apply this concepts to art.

Assistants were from very differents fields: electronic arts undergraduates, contemporary artists, electroacustic music composers but also physics and computer science undergraduates, graduates and professors.

In the third day we used AudioStellar to introduce topics of unsupervised learning (dimensionality reduction and clustering but also bias and fairness), audio features and signal processing (STFT, spectral centroid, visualization), jupyter notebook and python libraries numpy, librosa, matplotlib and sklearn.

At the end of the course students presented work-in-progress pieces at the WIP festival [6] we hosted with other teachers.

4.0.2 Workshop at DuraznoConf

We gave a 4-hours workshop at DuraznoConf [7] in Uruguay where we introduced some of the concepts of the AI & Art course but in a very reduced manner. We recorded the assistants in a series of tasks involving saying their name, numbers, clapping and such and then we analyze that corpus using the software. Discussions around very basic characteristics of audio arised like envelope, timbre and pitch that were easily exemplify using the software.

At the end of the workshop we made a (quite noisy) notebook jam.

4.0.3 Talks

We gave a talk at *Museo de Arte Latinoamericano de Buenos Aires* for a general public. We provided intuitions on machine learning topics centered around the concept of models, latent space, bias and fairness. Assistants downloaded the software and could grasp some these ideas in a hands-on manner.

We also gave several talks at *Universidad de Tres de Febrero* for electronic arts undergraduates to encourage the use of the software and introduce some basic unsupervised learning concepts.

5. FUTURE WORK

We consider AudioStellar is still in an early development stage although completely usable and stable. We encourage artists and programmers to try it out and collaborate by giving feedback, writing code and uploading tutorials and examples.

We have started a series of video tutorials that will be available in YouTube and will first cover the basics for using the software but will also feature related topics for hacking it.

We are moving forward adding flexibility and usability by introducing the concept of *units*. Units will replace *modes* and will be similar to tracks in a DAW: every unit will have its own volume, effects, inputs and outputs and there will be a type of unit for each mode, for example the user will be able to have several particle swarms with different effects.

We are studying generative ML algorithms [21, 22] with the goal of generating samples in the empty spaces of the latent space.

We found that for our current pipeline audio clips lengths are very relevant. The result tends to cluster sounds together giving too much importance to the length, even when they are timbrically different. We are studying ways of encoding audio features to a fixed length vector that were successful in speech technology [23] in order to remove length from the criteria.

We would like to incorporate an easy-to-use tool for chopping large audios into smaller ones by a fixed length or by its onsets.

Acknowledgments

We would like to thank interns Sofia Efrón and Sabrina Garcia, whom have recently joined our research group, for their collaboration. Also, the whole MUNTREF Arte y Ciencia [8] core team for helping us disseminating our activities. Miguela García Pannelli for her support and graphic design. Last but not least, Mariano Sardón for making our research group a reality.

[5] https://i3a.gitlab.io/curso/

[6] http://ccmatienzo.com.ar/wp/events/wip-2/

[7] http://https://duraznoconf.uy

[8] http://arteyciencia.untref.edu.ar/

6. REFERENCES

[1] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[2] L. McInnes, J. Healy, N. Saul, and L. Grossberger, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *The Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018, arXiv: 1802.03426. [Online]. Available: http://arxiv.org/abs/1802.03426

[3] M. Casey, "Soundspotting: A New Kind of Process?" in *The Oxford Handbook of Computer Music*, Apr. 2011.

[4] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton, "Real-Time Corpus-Based Concatenative Synthesis with CataRT," *9th International Conference on Digital Audio Effects (DAFx)*, pp. pp.279–282, 2006.

[5] G. Roma, O. Green, and P. A. Tremblay, "Adaptive Mapping of Sound Collections for Data-driven Musical Interfaces," in *NIME'19*, 2019.

[6] M. Tan and K. McDonald, "The Infinite Drum Machine | Experiments with Google." [Online]. Available: https://experiments.withgoogle.com/drum-machine

[7] L. H. Hantrakul, "lamtharnhantrakul/klustr," Aug. 2019, original-date: 2017-12-07T03:11:55Z. [Online]. Available: https://github.com/lamtharnhantrakul/klustr

[8] "XO - XLN Audio." [Online]. Available: https://www.xlnaudio.com/products/xo

[9] R. Fiebrink, D. Trueman, and P. Cook, "A Meta-Instrument for Interactive, On-the-Fly Machine Learning," in *NIME09*, 2009.

[10] M. Schedel, P. Perry, and R. Fiebrink, "Wekinating 000000Swan : Using Machine Learning to Create and Control Complex Artistic Systems," 2011.

[11] M. Schedel and R. Fiebrink, "A Demonstration of Bow Articulation Recognition with Wekinator and K-Bow," 2011.

[12] R. Fiebrink, "Machine Learning Education for Artists, Musicians, and Other Creative Practitioners," *ACM Transactions on Computing Education*, 2019.

[13] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," *arXiv:1812.04948 [cs, stat]*, Mar. 2019, arXiv: 1812.04948. [Online]. Available: http://arxiv.org/abs/1812.04948

[14] M. Grierson, M. Akten, and R. Fiebrink, "Deep Meditations: Controlled navigation of latent space," in *32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.*, 2018.

[15] J. Elwes, "Latent Space." [Online]. Available: https://www.jakeelwes.com/project-latentSpace.html

[16] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *WIREs Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.30

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[18] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors, "SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python," *arXiv:1907.10121 [physics]*, Jul. 2019, arXiv: 1907.10121. [Online]. Available: http://arxiv.org/abs/1907.10121

[19] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and Music Signal Analysis in Python," in *13th Python in science conference*, 2015.

[20] "PyInstaller." [Online]. Available: https://www.pyinstaller.org/

[21] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial Neural Audio Synthesis," in *ICLR 2019*, 2019, p. 17.

[22] A. Marafioti, N. Holighaus, N. Perraudin, and P. Majdak, "Adversarial Generation of Time-Frequency Features with application in audio synthesis," in *36th International Conference on Machine Learning*, Feb. 2019.

[23] Y.-A. Chung, C.-C. Wu, C.-H. Shen, H.-Y. Lee, and L.-S. Lee, "Audio Word2Vec: Unsupervised Learning of Audio Segment Representations using Sequence-to-sequence Autoencoder," Sep. 2016.