Proceedings of the International Conference on Live Coding, Valdivia, December 2021
https://iclc.toplap.org/2021/

Live Coding and Machine Learning is Dangerous: Show us your Algorithms

Shelly Knotts
Independent Researcher
shelly@datamusician.net

Iván Paz
Independent Researcher *on-the-fly*
ivanpaz@upc.edu

ABSTRACT

Machine learning encompasses computer algorithms able to learn through experience and by using data, with the primary aim of the optimization of automation processes. In integrating machine learning with live coding questions are raised around: what to optimize; which processes to automate; the role of the performer; and how to present ML algorithms and processes to the audience. Although there are no strict boundaries or limits in the practice of live coding, the TOPLAP draft manifesto has often provided a guiding philosophy for the development of practices. In combining live coding with ML, we find some tensions and frictions in staying true to the concepts presented in the manifesto. Although the authors do not advocate puritanism, in considering the combination of live coding and ML from the lens of the TOPLAP draft manifesto, we find an axis on which to generate discussion contemplating the limits, potentials and challenges in combining two technological practices which at times seem to sit at opposite ends of the spectrum when it comes to core live coding principles such as transparency, liveness, and legible processes.

1. INTRODUCTION

Machine learning encompasses computer algorithms able to learn through experience and by using data, with the primary aim of the optimization and automation of processes. Recent philosophical thought in the field of ML has raised issues around data and algorithmic bias; transparency; authorship (Raji and Buolamwini 2019; Buolamwini and Gebru 2018; D'ignazio and Klein 2020; Bellardrini et al. 2019); what to optimize; which processes can, and should be automated (Broussard 2018; O'Niel 2016). These issues warrant exploration in integrating machine learning (ML) with live coding alongside domain specific questions around the role of the performer and how to present ML algorithms to the audience. Although there are no strict boundaries or limits in the practice of live coding, the TOPLAP draft manifesto (2004), has often provided a guiding philosophy for the development of practices. In combining live coding with ML, we find some tensions and frictions in staying true to the concepts presented in the manifesto. Although the authors do not advocate puritanism, in considering the combination of live coding and ML from the lens of the TOPLAP draft manifesto, we find an axis on which to generate discussion. We contemplate the limits, potentials and challenges in combining two technological practices which at times seem to sit at opposite ends of the spectrum when it comes to core live coding principles such as transparency, liveness, and legible processes.

As the use of ML in live coding performance is growing within the community (see for example: Xambo 2021; Kiefer and Magnusson 2019), we see an opportunity to discuss the technical challenges and aesthetic decisions that may be suggested by such systems. A variety of approaches have been taken to combining ML with live coding, and in this paper, we use the manifesto as a dimension through which these can be analyzed, focussing on musical and sound

applications as this medium is of particular interest to the authors. We explore how new practices and technologies may present a challenge to the longevity of the manifesto as hardware advances widen the scope of processes that can be computed in real-time, and the still existent challenges to a truly live coded ML performance.

Sometimes we find friction in combining the technical approaches to the two fields. Live coding promotes performance transparency. That is, it aims to visualize algorithmic processes for an audience. When combining live coding with ML, we can reflect on how ML algorithms could be visualized, and if this is desirable. It goes without saying that live coding systems are designed for real time code compilation, however, carrying out ML processes in real-time compromises the cognitive complexity of the algorithms, as performers need to operate them on stage, or the size of the data sets, which may make the training times more or less suitable to live performance. On a technical level, at the performative moment, live coding and ML operate at different levels of abstraction, with live coding often being analogous to preparing homemade pizza from scratch v.s. ML which is more akin to calling your favorite place and choosing l'impasto della pizza and the ingredients. The transparency of live coding brings with it philosophical considerations around authorship and IP which run contrary to normal processes in the music industry, however with ML we find deeper reflection needed given the frequent reliance of ML algorithms of large, largely uncredited data sets.

In our analysis we find three main areas of consideration when working with live coding and ML and below unpick some questions and potentials of aesthetic exploration in these categories. These are Real Time Machine Learning, including technical and performative aspects, Data, including issues dataset size, authorship and transparency and Visualization of processes and algorithms.

We finish the document with a brief (although far from comprehensive) exploration of projects which use a combination of ML and live coding and propose some initial categorisations which appear to be generating the most aesthetic exploration in this hybrid space.

1.1 The Manifesto

It is useful at this juncture to review the key tenets of the TOPLAP draft manifesto, from which we draw our title, and which acts as an axis upon which to center our discussion. Here is a fragment:

We demand:

- *Give us access to the performer's mind, to the whole human instrument.*
- *Obscurantism is dangerous. Show us your screens*
- *Code should be seen as well as heard, underlying algorithms viewed as well as their visual outcome.*
- *Live coding is not about tools. Algorithms are thoughts. Chainsaws are tools. That's why algorithms are sometimes harder to notice than chainsaws.*

We recognise continuums of interaction and profundity, but prefer:

- *Insight into algorithms*
- *The skillful extemporisation of algorithm as an expressive/impressive display of mental dexterity*

In light of the manifesto, the use of ML algorithms has implications in live coding practice. In the manifesto we find a dedication within the practice to contemplating, and visualizing human and machine processes in performative and transparent ways. In typical ML processes, black box algorithms are often used, where even the programmer is not able to unpick the mapping between input and output, clearly challenging the live coding community’s dedication to transparency.

Although some ML processes are more legible, these may still present challenges to the audience in unpicking what is coded in real time and what is the result of algorithmic decision making.

The liveness aspect of live coding is also challenged by ML algorithms which may take longer than performance time to train, and in which abstracting low level tasks into algorithmic decision making processes changes the role of the performer in complex ways.

We interrogate these issues in the sections that follow.

2. QUESTIONS/TOPICS

2.1 Real Time Machine Learning

One approach to combining live coding and ML is to attempt real time machine learning. That is, to attempt to train models as a part of a live performance. While this maintains the liveness and performativity of live coding, by integrating the ML algorithms into the human-algorithm interaction loop, it brings with it technical challenges relating to computational resources, and time constraints on training loops.

2.1.1 Technical Challenges

Real time machine learning has technical implications related to the size of the datasets given the computational complexity of the algorithms. For example, the training complexity of nonlinear Support Vector Machines, although depending on several factors, lies, in general, between $O(n^2)$ and $O(n^3)$, with n being the amount of training instances (Abdiansah and Wardoyo 2015). This creates a technical challenge if the dataset is large. Also, some algorithms have many parameters which are difficult to configure in real time. For example, the Multi Layer Perceptron Classifier in python (for a system using sklearn library see: Diapoulis 2017) has 23 parameters. While it is not necessary to change all the parameters each time, without a proper mapping or expertise, the relationship between the parameter values and the aesthetic result is not always clear.

The Flucoma (Green et al. 2019) project, aims to provide “potent algorithms with a suitable level of modularity within the main coding environments used by the creative researchers”. To achieve this aim, Flucoma’s implementation of the FluidMLPClassifier has just 8 parameters to configure. Scaling back the parameter space allows the classifier more flexibility to run at performance time and reduces the complexity of mapping. Flucoma’s algorithm is designed with a workflow which can be used for purposes such as live coding. However, there is still a time constraint related to finding a proper mapping or exploring which parameters provide the best affordances for creative use, which makes these algorithms difficult to use within a real-time context.

2.1.2 The Role of the Performer

Despite the technical challenges of engaging with ML training processes in real time, live coding still offers opportunities to explore ML from a performative perspective. In most musical ML contexts, a musician would use a pretrained model to generate their creative outputs. This process involves deciding the parameters to automate, training the model, followed by testing, and optimisation processes, where speed, efficiency and accuracy are prioritized. Unlike with live coding, changing mappings is a lengthy process involving iterative training of the model.

In combining ML with live coding a pre-trained model could be used, however, a more radical approach is to eschew optimisation all together, and to train a model as a performance such as in Baalman’s GeCoLa (Baalman 2020). In GeCoLa, the performer trains a neural net in real time to recognise physical gestures and maps sounds in response to them. Baalman’s performance uses the live coding paradigm of performative interaction between human and algorithm to highlight the laborious, repetitive and error prone process of training a ML model. Although this process may not

conform to the manifesto's call to reveal underlying algorithms, as the neural net itself is not visible to the audience, the protraction of the training process performatively portrays the complexity of the algorithmic mapping process. This may seem to be in contravention to the philosophical justifications of ML in abstracting and automating processes, and in doing so, reducing human workloads and decision making, however, non-optimised ML algorithms do leave much space for creative exploration.

2.1.3 Balancing Training and Risk

Anna Xambó's MIRCLa (Music Information Retrieval Live Coding Agent) (Xambó 2021), aims to explore big collections of sound data in live coding performance by means of ML and music information retrieval algorithms. The live coder trains a virtual agent that can manage tasks such as retrieving similar sounds during the performance. The retrieved sound can then be processed, sequenced, etc, on-the-fly. The agent is generally trained offline, although the technical implementation would allow it to be trained as a performance. The performers are able to control the level of training of the agent, which gives creative scope for the performers to explore aspects of surprise and risk in the performance. For instance a highly trained agent would return more predictable results than an agent with less training. Live coding practices are echoed here in the flexibility of the training process, which could be seen as analogous to the practices of blank slate and full slate, in which the performer's level of code preparation to some degree affects the level of risk they are taking in the performance.

2.2 Data

2.2.1 Transparency, Authorship and IP

Live coding challenges traditional narratives of ownership and IP in music (Zeilinger 2014). When showing the screen a performer shares the classes, functions, configurations, etc. Through the practice live coders often learn new tricks and techniques from watching someone else performing. Together with the wide availability of free and open source software for live coding and the multiple channels and modes of information exchange that exist within the community, this challenges traditional conceptions of creative production where someone owns the rights to the final output of the work. At /*vivo*/ festival held in México City in 2012 (/*vivo*/ 2012a; 2012b), the final versions of the code after each session were placed in a shared folder as an invitation to share, read and rewrite each other's code. Sharing the code has continued to be an important practice among the community. For example, Alex McLean's Peak Cut EP (McLean, 2015) Paz's Visions of Space (Paz, 2017) both included executable and editable code releases, significantly subverting standard practices of music production and IP. The code and music becomes data that is easily shared, remixed and reworked.

Where live coding meets ML these narratives of transparency, anti-art and authorship become complex, particularly relating to the use of data intensive models. Authorship is a contested zone in the field of ML (McCormack et al. 2019; Ruipérez et al. 2017). Given the vast amount of data and server time required to train a musical model to a sufficient degree, emerging practices often use pre-trained models such as Google's Magenta algorithms (Huang and Raymond 2016). In these cases the origins of produced material are largely uncredited to the data sources. Feminist theories around data and AI (Leavy et al. 2021; Williams 2018; Cinders 2020) ask us to consider the impacts of data bias and how this may affect outputs and aspects of cultural production. This critique feels particularly relevant in the field of live coding and ML given live coding's dedication to transparency. If we use models with untraceable training data, we could ask ourselves what biases we are coding into the fabric of the live coding aesthetic? How can we claim to 'have access to the whole human

instrument' if part of that instrument is generated through stacks of inaccessible and uncredited data? (Grund 2005; Holzapfel et al. 2018)

Some approaches to musical ML use the artists own back catalog of musical data to train models, both for reasons of authorship contestation and for stylistic consistency across the artists own repertoire. In this case, issues around data origins and transparency feel less problematic, given that outputs are routed in the artist's own practice. In the next sections we discuss some systems that use such small datasets for live coding performance.

2.2.2 Small vs. Big Data

It is possible to integrate algorithms pre-trained over large datasets into live coding systems, for low level creative tasks such as in melody generation. SeMA (Bernado et al. 2020), for example, is a playground for prototyping live coding mini-languages that integrates signal synthesis and ML. It facilitates the use of models from TensorFlow (Abadi et al. 2016), with live coded mapping of inputs and outputs. The training time of some models can take minutes, hours or days depending on the algorithm, the dataset, and the hardware. Once trained, TensorFlow models can be fairly accurate, rarely surpassed by other systems. This big data approach reaches philosophical limits like the one suggested by Collins (2016) — an algorithm can be trained over corpora of music that would take a human years to listen through.

Some live coding artists have eschewed such large-scale datasets for smaller datasets which can be highly personalized and offer more flexibility of interaction. Niklas Reppel opened his presentation at the fifth International Conference on Live Coding (Limerick 2020) with the question: Why small data? Stating the ideas behind the design of Megra, a mini-language to make music with variable-order Markov chains (Reppel 2020) he answered in the following way: Small data is a defined response to the current algorithms, which are mostly focussed on big data sets and specialized hardware, and therefore have long training times that won't fit into the live coding performance. Working with small datasets can have it's limits however, with Reppel stating: "Megra is designed considering everything you can do with tiny datasets having real time feedback."

2.3 Visualizing Processes

As the TOPLAP manifesto's call "Obscurantism is dangerous. Show us your screens." makes clear, transparency of process is a key component of live coding practice. Much attention has been paid in the literature to the legibility of processes displayed through the projection of code windows and the subsequent impact of this aspect of live coding on practice and community (e.g. Roberts and Wakefield 2018; Knotts 2020; Burland and McLean 2016).

Combining live coding with ML requires a system designer to decide which algorithmic processes to abstract and to complete with algorithms instead of a human decision making process. Necessarily the mapping of the abstracted parameters are then obscured both to the audience and to the performer. It could be argued that the obfuscation of process breaks a key practice of live coding as it becomes impossible to reveal key aspects of the sound making process to the audience.

However, even in non-hybrid live coding contexts questions always remain on how the level of abstraction used in a live coding performance may reveal or obscure particular components of a performance. Languages designed for brevity and speed of typing such as IxiLang of TidalCycles for example may obscure more internal processes than lower level environments such as SuperCollider or Extempore. And on the contrary the complexity of the language design of SuperCollider makes understanding the relationship between code and sound more complicated than languages designed for readability such as Sonic Pi (Aaron 2016).

We could take this argument further, stating that in any live coding performance the processes of the human performer are only partially revealed through the individual coding moves made on stage. The decisioning processes in music performance are complex and include a vast quantity of multi-domain knowledge of the performer, alongside environmental factors in the moment of the performance. Previous attempts have been made (e.g. Knotts 2018; Knotts and Armitage 2018) to reveal some of these human processes through biometric sensing, but it's reasonable to state that the TOPLAP manifesto's call to 'Give us access to the performer's mind, to the whole human instrument' is by far the most ambitious project of the live coding scene! Further discussions on this point are beyond the scope of this article, but if we theorize the live coding performance as revealing the point of intersection between the performer's mind and the algorithmic system in play we can argue that understanding the full internal processes of ML algorithms is not essential to the live coding performance, but that being able to differentiate between that which is algorithmically performed by the human, and that which is defined through algorithmic decision making processes such as ML algorithms is important to avoid the danger of obscurantism highlighted in the manifesto.

This could be approached in numerous ways. For example, Xambo (2021) states: "Emphasizing the legibility of the code is relevant so that it is clear at all times the processes and decisions taken by both the VA live coder and also the human live coder. It is still an open question how to find the right balance between simplicity and complexity."

From this we could conceptualize that languages with a greater degree of clarity in the code-to-sound relationship would allow performer and audience members to understand that which is created through in-the-moment human action and that which is generated through ML processes. SeMA (Bernado et al. 2020) e.g. has a clear interface at the point that data is streamed to or from a ML algorithm.

The call to 'show us your screens' has driven some artists working with ML to attempt to visualize aspects of the ML algorithms in order to more fully express the process of their performances to audience members. Scorpion Mouse's t-SNE maps (Levine 2018) are used to traverse related sonic spaces, and Levine also provides a visualization of this process in performance.

This brings with it questions on how to visually represent the algorithmic processes (McLean 2010). Using other modalities i.e. visuals when doing a music performance provides other sensorial avenues for expressing the ML processes. However, this sometimes adds complexity for the audience rather than reducing it by requiring more of their focussed attention to understand the algorithmic processes at play.

Renick Bell's development process for his Conductive system (Bell 2013) includes abstracting and automating processes which are repeated often in a performance, continuously morphing his role as a performer to an ever higher level of abstraction. Renick's interface shows in text form that agents are performing low level tasks without revealing the details. This could be enough to communicate the conceptual direction of the performance and the role of the performer.

3. MODELS

The technical challenges and aesthetic frictions of combining live coding and ML has to date limited exploration in this area, although growing consciousness of techniques and availability of tools has expanded experimentation in recent years. While most early live coding performances focussed on low level tasks such as simple sound generators and processors and pattern writing (Villaseñor and Paz, 2020), today it is increasingly common to hear performances using ML to perform specific tasks. We have identified a number of models where live coding and ML find the most confluence, and which seem to have generated artistic exploration at the intersections of live coding and ML. We

highlight some systems under each category, however, this is by no means a comprehensive list of current exploration.

3.1 Agents as Collaborators

Renick Bell (2011; 2013) automates aspects that he finds himself repeating across multiple performances, leaving himself free to play with other aspects that are more malleable and less routine. His system Conductive uses a set of agents which each act as an instrument with a set of rhythmic patterns to choose from. These 'players' generate their own music autonomously. The performer acts as a conductor making high-level decisions such as rhythmic density and can turn players on and off manually. This enables Bell to 'manage' the performance without worrying about low-level details, instead focussing on the flow of the music overall. Conductive contains abstractions for musical time, musical events, and event loops. These abstractions allow the agents to operate as sequencers with some predefined musical properties, making them more intuitive from a musical perspective.

MIRLCa (Music Information Retrieval Live Coding Agent) (Xambo 2021) aims to explore the use of big collections of sound data in live coding performance by means of ML and music information retrieval algorithms. The live coder trains one or more virtual agents that can manage tasks such as retrieving similar sounds during the performance. The central ideas behind MIRLCa are legibility, agency and negotiability during the performance which are explored through the collaboration between the live coders (human and agent) and the audience.

Megra (Reppel 2019) is a mini-language to make music based on variable-order Markov chains. Among other algorithms, it includes a simplified version of the one proposed in (Ron et al. 1996). The algorithm learns probabilistic automatas with variable memory length from sequences. The Markov processes of variable memory length can be described by a subclass of probabilistic finite automata (which are the resulting models learned by the algorithm). The algorithm is efficient in terms of computing cost, and it exhibits good accuracy. Its authors proved that the KL-divergence between the target distribution and the learned distribution can be made small in polynomial time. In the Megra implementation, the input is a string of characters written by the performer on stage (e.g xoxxo----xxo) that is analyzed in real time to learn the probability of each symbol. The probability of each character is mapped to specific samples or synths. The whole process: data collection, learning and the production of the output, is performed in real-time.

3.2 Autonomous Live Coding Systems

Although not a fully fledged neural net IxiLang (Magnusson 2011) provides the autocode function which generates randomized executable lines of code which can then be edited by the performer.

Cibo (Stewart et al. 2020) is an autonomous performer that uses neural networks, and gave its first performance as a solo artist with a set at the International Conference of Live Coding 2019. Cibo takes only code as an input and is trained on sequential code blocks captured from human performed TidalCycles sets. The authors mention that Cibo is trained only with code, without having access to sound signals.

Cacharpo (Navarro and Ogborn 2017) reduces the task of automatic music making to a single tightly defined genre of Cumbia Sonidera. The autonomous performer uses Music Information Retrieval (MIR) algorithms to listen to the human performer and generates code which produces complementary patterns and instruments. Cacharpo uses the metaphor of an assistant of a driver in some Mexican micro-buses (called Cacharpos) who are said to aspire to having their own bus.

3.3 Performative Model Training

Marije Baalman's (2020) GeCoLa project which performatively trains a neural net to recognise physical gestures and map sounds in response to them.

3.4 Environments

SEMA (Bernado et al. 2020) is an interface, which aims to allow easy integration of a customisable live coding language with Magenta's ML plug-ins which can themselves be edited during performance. The system includes two windows— one for live coding, and another containing code to interact with ML algorithms, and some simple syntax to pass data to and from the ML algorithms. This allows the performer to freely pass sequences of data from the live coding window to the input of the ML model and receive new sequences back according to the model.

3.5 Pre-labelled Data for Audio Engines

RuLer is a rule learning algorithm designed to perform induction, creating new data, out of a label dataset (Paz et al. 2020). The degree of consistency with the original material is 'controlled' by two parameters. These define: the allowed distance between examples from which it is possible to create new data; and the percentage of original data that needs to be contained in a rule to be created. The algorithm was designed with only two parameters in order to facilitate its use in real time. The system was designed to automatically produce variations from a set of labeled synthesizer presets during a live coding performance. The material can be selected before or mid-performance, and the degree of consistency is used to create tensions and variations while the performer conducts the high level evolution of the sound.

4. CONCLUSIONS

This contribution proposes some implications of using ML within live coding performance, contextualizing them from the perspective of the TOPLAP draft manifesto. We observe how different applications visualize different abstraction levels and moments of the ML process. Acknowledging such conditions and consequences allows us to pay attention to how the decisions we make might shape our creative outputs in different directions. This reflection is important since live coding is about human interaction with algorithms, which are bound by technical restrictions when it comes to ML.

This condition, of course, is not inherent to ML, for example, different languages allow different live coding tasks to be foregrounded and perceived. For example, the primacy of sound synthesis in SuperCollider or pattern writing in TidalCycles. This follows from the way that technology inherently shapes our performances. When we say "Live coding and Machine Learning is dangerous: Show us your algorithms", we both present an extreme case of friction between the technological approaches of ML and live coding, and reflect on the consequences of simultaneously engaging with the different approaches.

When using ML within live coding, we find two technological models that may shape our creative outputs in opposite directions, but significant creative exploration exists at the intersection. These directions, in general, are shaped by the trade-off between having highly accurate results v.s. having real-time feedback from the learning algorithm. Somewhere close to the intersection lie unruly algorithms where real-time inaccuracies may need to be embraced and harnessed by the performer, in the same way the live coder embraces coding errors as part of the performance process.

The range of systems encompasses virtual collaborators and fully autonomous agents. Although some systems include models trained on large datasets, we identify a trend in the design and

implementation of algorithms focused on small data, sometimes as a response to the dependency of big data deep learning models on expensive hardware. Some datasets are tagged directly by the live coders while others are collected over large periods of time, for example during different performances. In some cases, algorithms are also designed for a specific task. Such approaches seek real-time interaction with the learning algorithm, making it an integral part of the performance. These algorithms generally have fewer parameters, which are designed to produce outputs with clearer mappings to the inputs, producing greater scope for expressivity.

We can reflect that ML is largely about optimization, but what do we want to optimize in live coding? One approach suggests a move to higher-level performances where perhaps the production of low-level material is carried out by trained algorithms, in the same way that we can use sound generator units during a performance without worrying about writing the synthesis.

As the digital nature of live coding brings algorithmic possibilities, new instruments and environments centered on them are emerging alongside new technologies. Combining these systems with ML presents the possibility for these artifacts to change by interacting with the performer, or to perceive and adapt to the external reality by means of machine listening.

The approaches discussed define extreme possibilities. Each has to embrace its necessary consequences. However, just as technologies condition our way of making music, live coding, which finds its guiding principles in the TOPLAP draft manifesto is shaping technological developments, designing systems that learn in real time, exploring the limits and edges of ML algorithms from a creative perspective, analyzing how they change through different training datasets and writing new ones that provide desired affordances.

REFERENCES

Aaron, Sam. "Sonic Pi–performance in education, technology and art." International Journal of Performance Arts and Digital Media 12.2 (2016): 171-178.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M.,. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16), pp. 265-283. 2016.

Abdiansah, Abdiansah, and Retantyo Wardoyo. "Time complexity analysis of support vector machines (SVM) in LibSVM." International journal computer and application 128, no. 3 (2015): 28-34.

Baalman, Marije. "The machine is learning." International Conference on Live Interfaces, 2020.

Bell, Renick. "An interface for realtime music using interpreted haskell." *Proceedings of LAC 2011* (2011).

Bell, Renick. "An Approach to Live Algorithmic Composition using Conductive." In *Linux Audio Conference 2013*, p. 29. 2013.

Ballardini, Rosa Maria, Kan He, and Teemu Roos. "AI-generated content: authorship and inventorship in the age of artificial intelligence." In Online Distribution of Content in the EU. Edward Elgar Publishing, 2019.

Bernardo, Francisco, Chris Kiefer, and Thor Magnusson. "A Signal Engine for a Live Coding Language Ecosystem." Journal of the Audio Engineering Society 68.10 (2020): 756-766.

Broussard, Meredith. Artificial unintelligence: How computers misunderstand the world. MIT Press, 2018.

Burland, Karen, and Alex McLean. "Understanding live coding events." International Journal of Performance Arts and Digital Media 12.2 (2016): 139-151.

Buolamwini, Joy, and Timnit Gebru. "Gender shades: Intersectional accuracy disparities in commercial gender classification." In *Conference on fairness, accountability and transparency*, pp. 77-91. PMLR, 2018.

Sinders, Caroline. "Feminist Data Set." (2020).

Collins, Nick. "Towards machine musicians who have listened to more music than us: Audio database-led algorithmic criticism for automatic composition and live concert systems." Computers in Entertainment (CIE) 14.3 (2016): 1-14.

D'ignazio, Catherine, and Lauren F. Klein. Data feminism. MIT press, 2020.

Diapoulis Georgios. “Live Coding using SC3 and scikit-learn”. November 10, 2021. http://gewhere.github.io/blog/2017/10/13/live-coding-using-sc3-and-scikit-learn/

Green, Owen, Pierre Alexandre Tremblay, and Gerard Roma. "Interdisciplinary Research as Musical Experimentation: A case study in musicianly approaches to sound corpora." In *Electroacoustic Music Studies Network Conference: Electroacoustic Music: Is it Still a Form of Experimental Music?*. Zenodo, 2019.

Grund, Cynthia M. "Music Information Retrieval, Memory and Culture: Some Philosohpical Remarks." In *ISMIR*, pp. 8-12. 2005.

Holzapfel, Andre, Bob Sturm, and Mark Coeckelbergh. "Ethical dimensions of music information retrieval technology." *Transactions of the International Society for Music Information Retrieval* 1, no. 1 (2018): 44-55.

Huang, Allen, and Raymond Wu. "Deep learning for music." *arXiv preprint arXiv:1606.04930* (2016).

Kiefer, Chris, and Thor Magnusson. "Live coding machine learning and machine listening: a survey on the design of languages and environments for live coding." Proceedings of the International Conference on Live Coding, Media Lab, Madrid, Espagne. 2019.

Knotts, Shelly. "Social Systems for Improvisation in Live Computer Music." PhD diss., Durham University, 2018.

Knotts, Shelly. "Live coding and failure." The Aesthetics of Imperfection in Music and the Arts: Spontaneity, Flaws and the Unfinished (2020): 189.

Knotts, Shelly, and Joanne Armitage. “VIBEZ: A Small Sense of Presence at a Distance.” Proceedings of International Conference on Live Interfaces, Porto. 2018.

Leavy, Susan, Eugenia Siapera, and Barry O'Sullivan. "Ethical Data Curation for AI: An Approach based on Feminist Epistemology and Critical Theories of Race." Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society. 2021.

Levine, Jason. “Combining Livecoding and Real-time Software for Musical Improvisation, Arts at MIT”. (November 17, 2021. https://www.youtube.com/watch?v=27ZslA6_vXE

Magnusson, Thor. "The ixi lang: A supercollider parasite for live coding." In ICMC. 2011.

McCormack, Jon, Toby Gifford, and Patrick Hutchings. "Autonomy, authenticity, authorship and intention in computer generated art." International conference on computational intelligence in music, sound, art and design (part of EvoStar). Springer, Cham, 2019.

McLean, Alex, et al. "Visualisation of live code." Electronic Visualisation and the Arts (EVA 2010) (2010): 26-30.

McLean, Alex. “Peak Cut, 2015 Computer Club.”. November 5, 2021. https://computerclub.bandcamp.com/album/peak-cut

Navarro, Luis, and David Ogborn. "Cacharpo: Co-performing Cumbia Sonidera with Deep Abstractions." Proceedings of the International Conference on Live Coding. 2017.

O'neil, Cathy. Weapons of math destruction: How big data increases inequality and threatens democracy. Crown, 2016.

Paz, Iván. “Visions of Space, 2017”. November 5, 2021. https://bohemiandrips.bandcamp.com/album/visions-of-space

Paz, Iván, Àngela Nebot, Francisco Mugica, and Enrique Romero. "On-The-Fly Syntheziser Programming with Fuzzy Rule Learning." Entropy 22, no. 9 (2020): 969.

Raji, Inioluwa Deborah, and Joy Buolamwini. "Actionable auditing: Investigating the impact of publicly naming biased performance results of commercial ai products." In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 429-435. 2019.

Reppel, Niklas. “Megra International Conference on Live Coding, 2020”. November 5, 2021. https://youtu.be/6dhvNrwQTRU?t=3839.

Roberts, Charlie, and Graham Wakefield. "Tensions and Techniques in Live Coding Performance." (2018): 293-317.

Ron, Dana, Yoram Singer, and Naftali Tishby. "The power of amnesia: Learning probabilistic automata with variable memory length." Machine learning 25.2 (1996): 117-149.

Ruipérez, C., E. Gutiérrez, Cristina Puente, and J. A. Olivas. "New Challenges of Copyright Authorship in AI." In *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, pp. 291-296. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2017.

Stewart, Jeremy, and Shawn Lawson. "Cibo v2: Realtime Livecoding AI Agent." Proceedings of the 2020 International Conference on Live Coding (ICLC2020). 2020.

TOPLAP “Manifesto draft, 2004”. November 20, 2021. https://toplap.org/wiki/ManifestoDraft

Villasenor, Hernani and Ivan Paz. “”Live coding from scratch: the cases of practice in Mexico City and Barcelona””. In: International Conference on Live Coding. 2020.

/*vivo*/ 2012a “Simposio Internacional de Música y Código Mexico City” October 15, 2021. https://www.flickr.com/photos/90948585@N07/

/*vivo*/ 2012b. “Simposio Internacional de Música y Código Mexico City” October 11, 2021. https://web.archive.org/web/20140406043458/http://vivo2012.cenart.tv/

Williams, Betsy Anne, Catherine F. Brooks, and Yotam Shmargad. "How algorithms discriminate based on data they lack: Challenges, solutions, and policy implications." Journal of Information Policy 8 (2018): 78-115.

Xambó, Anna. "Virtual Agents in Live Coding: A Short Review." arXiv preprint arXiv:2106.14835 (2021).

Xambó, Anna. "Live Coding with Crowdsourced Sounds and A Virtual Agent Companion." (2021).

Zeilinger, Martin. "Live coding the law: Improvisation, code, and copyright." Computer Music Journal 38.1 (2014): 77-89.