Building a dataset of personal live coding style using Mirlcaproxy: a journal of creative sonic exploration under constraints and biases

Building a Dataset of Personal Live Coding Style Using MIRLCaProxy: A Journal of Creative Sonic Exploration under Constraints and Biases

Anna Xambó Sedó
Centre for Digital Music,
Queen Mary University of London
a.xambosedo@qmul.ac.uk

Gerard Roma
University of West London
gerard.roma@uwl.ac.uk

ABSTRACT

This paper presents the technical and creative process of building a dataset of a personal live coding style using MIRLCaProxy, a custom SuperCollider class built on top of the MIRLCa extension. MIRLCa enables real-time sampling of sounds from Freesound with the assistance of machine learning using FluCoMa. We designed an environment that captures eight methods for retrieving sounds in the MIRLCa language, recorded through a live coding journaling approach. This approach aims to predict the next line (next method) from the audio state of the system. Throughout the dataset creation, the required number of actions led to unexpected creative discoveries, transforming the process into a space for sonic exploration. This paper reflects on how the training of the machine learning process becomes a rehearsal space that supports the development of a personal style through constraints. It also explores the role of biases in this context.

1 Introduction

Live coding and digital sampling can be a creative approach to music making, especially when taking advantage of online crowd-sourced databases such as Freesound (Font, Roma, and Serra 2013). Using Freesound in live coding opens the sonic possibilities to sounds recorded under Creative Commons licences that can be repurposed algorithmically. In our previous research (Xambó 2023), we have seen the benefits of interactive machine learning (IML) (Fails and Olsen Jr 2003; Fiebrink and Caramiaux 2018) and working with small datasets in live coding when applied to train models with the custom-made system MIRLCa. The on-the-fly requested sounds from Freesound can be filtered with a machine learning classifier to obtain sounds with a greater control closer to what the live coder conceives as 'good' sounds (Xambó et al. 2021). In our previous research, we showed that using a binary classifier of 'good' versus 'bad' sounds was useful. However, it can also be limiting because it only acts as a selector. A virtual agent can offer more than that, such as helping the live coder to decide what code to write in response to a human live coder action.

With the assumption that we can expand the task of a binary classifier in live coding to other tasks that connect with learning a personal style of live coding, this paper addresses the following research question: *What can building a dataset of your personal live coding style tell us about our personal style of live coding?* Our approach is based on using the same custom-made live coding environment MIRLCa (Xambó et al. 2021; Xambó 2023) and respectively training a model employing a core of eight instance methods of the customised live coding language. The objective is to present a proof of concept to predict what the live coder can write as the next line of code. This prediction is based on training a model using several rehearsals and expects to consider the live coder's personal style.

Compared to the first machine-learning task developed with MIRLCa (Xambó et al. 2021), which was training a binary classifier to predict between 'good' and 'bad' sounds, here we present a classifier of eight classes. The order of magnitude of the dataset is distinctively different from 130 sounds (training set: 77%, test set: 23%) to 800 sounds (training set: 87.5%, test set: 12.5%) that covers eight classes to predict the next line of code. We share the lessons learned from the training process, which has become a journal of creative live coding explorations with the constraints given by the task. We present the process for creating an ongoing dataset as a space for creative exploration and reflection of the own personal style. This research also informs about the importance of maintaining the biases as characteristic of a personal style, which are difficult to learn by the learning algorithm. The approach undertaken can be seen as a promising sonic exploration to be part of the practice of a live coder.

2 Background

Several approaches have explored the integration of machine learning into live coding. While most systems conduct training offline prior to performance, some systems such as the Mégra system (Reppel 2020) brings the machine learning training process into the live coding performance itself. The ListeningLearning system is a machine improvisation system that performs with a human, which is developed throughout rehearsals and performances and combines machine listening and machine learning (Collins 2011). Sema (Bernardo, Kiefer, and Magnusson 2020) is an online platform that enables users to create custom live coding languages and adapt machine learning algorithms to their practice. IML can offer advantages to artists because of its ability to tailor software behaviour to individual needs through human-driven interaction. This presents both challenges and opportunities for exploring synergies between IML and live coding, particularly as both share the key property of *liveness* or "the ability to modify a running program'' (Tanimoto 2013, 31). Our work investigates these synergies by applying machine learning in the context of music performance. With the proposed MIRLCaProxy, the live coder can apply IML as part of the live coding practice. The live coder can rehearse for several sessions while building up the dataset using a live coding environment. This extensive time from multiple sessions is typically lacking in performance.

3 Method

3.1 MIRLCa

MIRLCa[1] (Xambó 2023) is a SuperCollider (SC) custom-made extension that inherits from the self-developed MIRLCRep2 SC extension (Xambó, Lerch, and Freeman 2018). The SC extension expands the MIRLCRep2's capabilities by proposing a virtual agent that embodies IML techniques. MIRLCRep2 has a range of performance methods that allow for querying sounds from Freesound based on human-made categories (folksonomy) or audio-based descriptors using the Freesound quark[2] and adding performativity characteristics with a constrained language. MIRLCa enables real-time sampling of sounds from Freesound with the assistance of machine learning using the Fluid Corpus Manipulation (FluCoMa) library (Tremblay, Roma, and Green 2021). Based on a binary classifier, the system predicts 'good' vs 'bad' sounds from Freesound. For this task, MIRLCa uses the class `FluidMLPClassifier`,[3] which performs classification using a Multi-Layer Perceptron (MLP) neural network (Xambó et al. 2021).

3.2 MIRLCaProxy

MIRLCaProxy aims to solve the choice of what function to use for the next live coding step, a decision based on the audio features of the current audio state of the system. This task intends to predict the next line of code. For this, we have designed a custom-made proxy class that allows for building the dataset using a live coding style.

Code	Method	Returned Freesound object	Argument
01	`.id`	A sound by its ID number	A Freesound ID number of a sound (`Integer`)
02	`.tag`	A sound by the requested tag (best candidate)	A tag (`String`)
03	`.similar`	A similar sound from the target (best candidate)	An index number of the target sound (`Integer`)
04	`.random`	A random sound (best candidate)	The number of returned sounds (`=1`) (`Integer`)
05	`.pitch`	A sound based on the pitch feature (best candidate)	The value of the feature in Hz (`Float/Integer`)
06	`.bpm`	A sound based on the bpm feature (best candidate)	The value of the feature in bpm (`Float/Integer`)
07	`.dur`	A sound based on the duration feature (best candidate)	The value of the feature in seconds (`Float/Integer`)
08	`.diss`	A sound by the dissonance feature (best candidate)	The value of the feature (`=[0..1]`) (`Float/Integer`)

Table 1: List of the eight instance methods and their constrained version in MIRLCaProxy to capture the style of a live coder.

`MIRLCaProxy` is a subclass of the MIRLCa class that captures eight methods for retrieving sounds from the MIRLCa language, recorded through a live coding journaling approach. This subclass allows the live coder to record their live coding sessions using a constrained subset of methods. For consistency, only eight methods with one parameter each are used as part of the query. A session consists of using the available methods in a live coding style, and the actions are recorded by the proxy class. The session is captured in a text file following the SC dictionary structure, which can

[1] https://github.com/axambo/MIRLCa
[2] https://github.com/g-roma/Freesound.sc
[3] https://learn.flucoma.org/reference/mlpclassifier

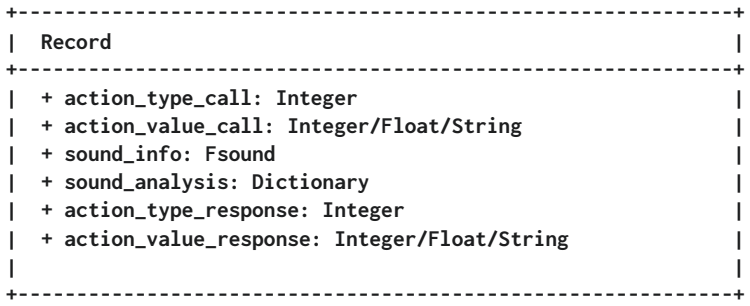

Figure 1: *A record stored in the dataset generated by the MIRLCaProxy subclass during a session. Diagram generated with Textik (https://textik.com/#d4e44b060ac3be2a)*

then be loaded again and used for training. Table 1 shows the list of the eight methods, the returned Freesound object and the type of expected argument to, overall, model and capture the sequence of actions.

MIRLCaProxy is a class used to build a dataset of entries that show call-response actions with values. This proxy class allows the creation of a dataset with encoded lines to predict the next line of code or next method. This can be used to train a model that aims to suggest, from a given query, what can be a suitable next line of code. Figure 1 shows the entity of a record in the dataset created in MIRLCaProxy during a session. The fields in each record of the dictionary provide:

- The type and value of the current action (`action_type_call`, `action_value_call`).
- The sound retrieved from the current action (`sound_info`).
- The sound analysis data from the current sound (`sound_analysis`).
- The type and value of the response action (`action_type_response`, `action_value_response`).

The following code example demonstrates how to instantiate the subclass, call the eight functions with their respective arguments, and save the session, as shown in the code editor:

```
a = MIRLCaProxy.new
a.id(9999)
a.tag("washing-machine")
a.random(1)
a.similar(0)
a.diss(0.5)
a.bpm(60)
a.dur(4)
a.pitch(100)
a.savedictionary
```

3.3 The Training Process

A repository containing all code used for data training and analysis, along with a selection of audio recordings from the sessions and related code discussed in the paper, is available at https://zenodo.org/records/15249330. The data discussed in this paper consists of 800 data points collected across 32 sessions. Each session, or rehearsal, contains 25 data points saved in a text file using the format described in Section 3.2. For each session (except the first), we also recorded the audio output in `.wav` format and saved the corresponding SuperCollider code for further analysis. In total, the recorded material amounts to 6 hours, 15 minutes, and 42 seconds. On average, each session lasted 12 minutes and 7 seconds, with the longest session lasting 16 minutes and 52 seconds, and the shortest 6 minutes. These durations reflect sustained musical engagement during the sessions.

We used the same MLP neural network and machine learning process as in the first task reported in Xambó et al. (2021). The 26 Mel Frequency Cepstral Coefficients (MFCCs) audio descriptors were reduced to 20 dimensions using Principal Component Analysis (PCA), and the resulting values were standardized to zero mean and unit variance. The final MLP architecture for this task consisted of a single hidden layer with 14 nodes and ReLU activations.

Figure 2 shows a diagram of the machine learning process flowchart. The follow-up explanation focuses on the classification of the eight methods presented here.[4] The key components of the machine learning process flowchart are:

[4]Future work includes applying the `FluidMLPRegressor` to predict the output values of the target method.

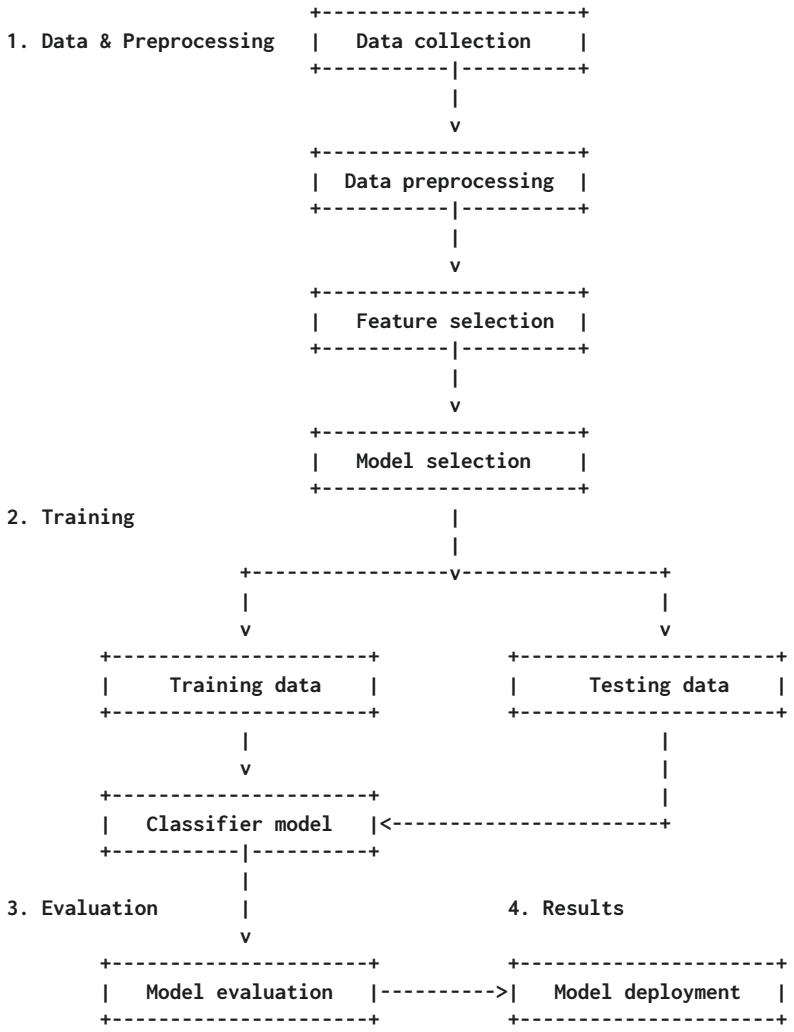

Figure 2: *A machine learning process flowchart of the different blocks of code used to train and predict the accuracy of the dataset. Diagram generated with Textik (https://textik.com/#626cc02b75693602)*

- **Data Collection**: This is the initial step where data is gathered from the text files. It includes text files that represent sessions. Each line is a data point.
- **Data Preprocessing**: The collected data is cleaned and transformed by unifying multiple files into a single dictionary. For training and testing, two `FluidDataset` and two `FluidLabelSet` objects are prepared, respectively.
- **Feature Selection**: MFCC features are selected as input data, while the `action_type_response` value is stored as the label and the sound ID as the dictionary key.
- **Model Selection**: The `FluidMLPClassifier` is selected, which performs classification between a `FluidDataSet` and a `FluidLabelSet` using a MLP neural network.
- **Model Training**: The small dataset (less than 1,000 data points) is split into a training dataset (87.5%) and a testing dataset (12.5%). The MLP model (`FluidMLPClassifier`) is trained using the preprocessed data, previously normalised using `FluidStandardize` (zero mean and unit variance) and `FluidPCA`, with the aim at feeding the data into the model and minimise error. `FluidMLPClassifier` performs classification between a `FluidDataSet` and a `FluidLabelSet` from the training dataset.
- **Model Evaluation**: After training, the model predicts labels for a `FluidDataSet` from the testing dataset. Model performance is evaluated using accuracy, defined as the proportion of correct predictions to total occurrences, to ensure the model generalises well to unseen data.
- **Model Deployment**: Once the model is validated, it can be deployed into a real-world environment where it can make predictions on new data.[5]

[5]While the results reported in this paper are promising, additional development and validation are required before the model can be deployed in practical settings.

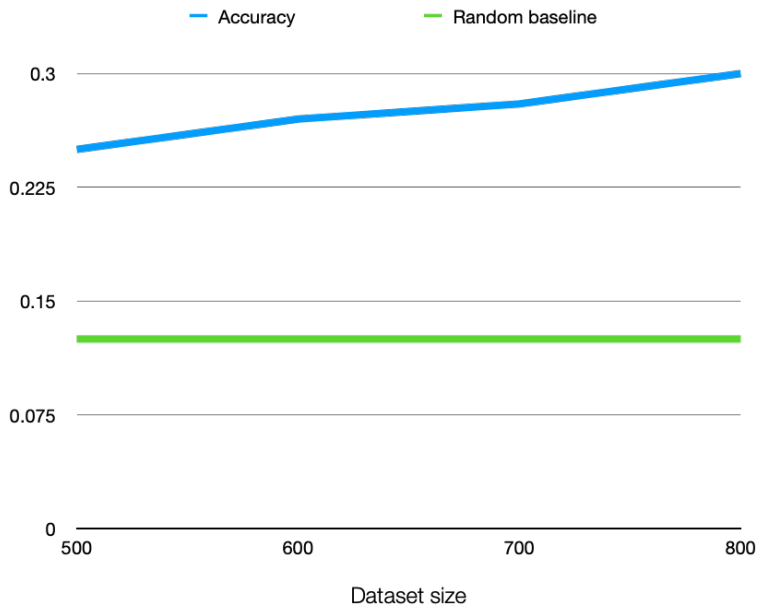

Figure 3: *A 2D line plot showing how accuracy increases with dataset size, compared to the random baseline.*

4 Findings

4.1 Technical Results

4.1.1 Accuracy Over Training Examples

To report accuracy, we trained the model across ten independent runs and selected the highest result. As shown in Figure 3, we found a slow improvement in the proportion of predicted classes as we increased the number of training examples from 500 to 800. Predictions were made using a set of 75 to 100 data points, representing 12.5% to 16.7% of the entire dataset.[6] Although the values are below 0.5, they should be compared to the random baseline of 0.125, which corresponds to the chance level for eight classes. The results do not indicate a consistent improvement as the number of training examples or sessions increases. While diminishing returns are expected when adding more training data to improve accuracy, additional data is still needed to determine when the system reaches saturation.

4.1.2 Confusion Matrix

We generated a confusion matrix to evaluate the performance of the classification model and understand how our model is making predictions by classes. The matrix was constructed using a sample of 100 predictions from the best-performing model trained on a dataset of 800 examples (achieving 30% accuracy with 700 training samples). This analysis aimed to identify the types of errors made by the model and to determine whether it is biased toward any particular class.

Figure 4 shows the result, which is a multi-class confusion matrix with eight classes. The diagonal elements represent correct predictions, while the off-diagonal elements indicate incorrect predictions. The class that performed best was `.similar` (11) followed by `.pitch` (7), `.tag` (5) and `.id` (5). By contrast, `.random`, (0) `.bpm`, (0), `.dur` (1) and `.diss` (1) were unpredictable.

We also calculated the performance metrics of accuracy, precision, recall, and the F1 score. The results presented below show low precision and recall for classes `random`, `bpm`, `dur`, and `diss`, indicating that the model may have difficulty identifying these classes with the current amount of training data:

- The **accuracy** was 30%, which we previously used as a criterion for model evaluation and is defined as the ratio of correct predictions to the total number of predictions. This value indicates that the model is not performing well. However, as discussed in the previous section, accuracy steadily improves with the addition of more training examples, as observed with a dataset size from 500 up to 800 data points.

[6]Because each session contained 25 data points, the training and testing sets were constructed in multiples of 25 for consistency and ease of use.

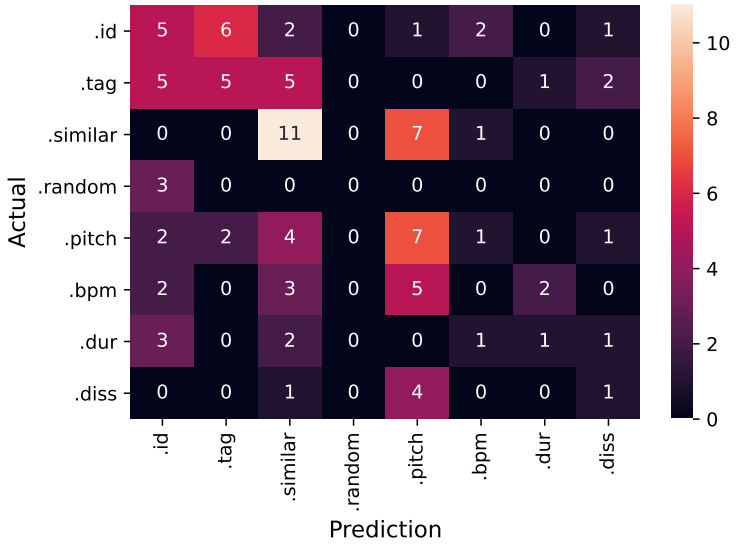

Figure 4: *A multi-class confusion matrix, where rows indicate actual classes and columns indicate predicted classes.*

- The **precision** concerns the ratio of true positive predictions to the total number of positive predictions. We obtained an overall score of 0.27 (weighted average) and the lowest values related to the classes `random` (0.0), `bpm` (0.0) and `diss` (0.17).
- The **recall** or sensitivity refers to the ratio of true positive predictions to the total number of actual positive instances (true positives + false negatives). We obtained an overall score of 0.3 (weighted average) and the lowest values related to the classes `random` (0.0), `bpm` (0.0), `dur` (0.12) and `diss` (0.17).
- The **F1 Score**, which is the harmonic mean of precision and recall (a combination of these two metrics into a single score), provides a balanced measure of model performance. We obtained an overall weighted average F1 score of 0.27, with the lowest scores observed for the classes `random` (0.0), `bpm` (0.0), `dur` (0.17), and `diss` (0.17).

4.2 Creative Results

The 32 sessions were conceived as improvisational rehearsals. The original constraints were considered as design constraints for composing and performing with digital musical systems (Magnusson 2010) with the hope that new musical possibilities could emerge. The nomenclature of each session has the following structure: `session_YYMMDD_HHMMSS`. The metadata of the Freesound sounds are captured in a credits text file to keep track of the authorship of the sounds.

To maintain efficiency, the first author, who recorded the sessions, avoided using any automatic or sound effect functions available in MIRLCa. As a consequence, the live coder is in the position of trying to find her voice in a highly constrained environment where the main intention is to tame the outcoming sounds to her particular style. Factoring out effects can be a limitation, which conditions the next step. Hence, the research presented in this paper should be seen as a proof of concept.

Throughout the dataset creation, the required number of actions led to unexpected creative discoveries, transforming the process into a space for sonic exploration under constraints and biases. We identified the following 'motifs' as recurring patterns present in the rehearsals: *the role of Float numbers, the meaning of similarity, exploration of randomness through number series* and *the role of repetition.* These are patterns present in the training that are non-mutually exclusive. On the contrary, the 'motifs' are often explored sequentially or combined.

4.2.1 The Role of `Float` Numbers

Using float numbers instead of integer numbers to define musical features such as pitch, duration and beats per minute (bpm) has resulted in leaning towards sound-based music (Landy 2007). For example, typing `.pitch(1256.6348)` instead of `.pitch(1256)` returns a soundscape-type sound instead of a melodic sound.

Some explorations of this concept involve using four-digit low decimal values for the `.diss` method in `session_240922_003003` or trying to find similarities among sounds by applying the pitch float value of one sound to retrieve another sound with the same float value in `session_240926_000248`. In `session_240921_001536` the concept of dissonance and out-of-tune sounds is explored using the same technique.

These sonic investigations align with the live coder's musical taste of avoiding melodies and finding musicality in the dissonances instead. Thus, the use of float values applied to melodic features such as pitch, duration or bpm can return sounds that may be preferable to build a sound-based music improvisation or composition.

4.2.2 The Meaning of Similarity

As shown in Section 4.1.2, `.similar` is the method best predicted by our model. To complement this method and explore the other available seven, the first author has investigated what the other methods tell us about similarity. This has entailed an exploration of nearby `.id` numbers that are below or up the current sound. For example, in `session_240919_235900`, the live coder discovers that the contiguous sounds in the Freesound database are likely to be similar:

```
b = MIRLCaProxy.new
b.random(1)
b.whatid // 369694
b.id(369693)
b.id(369695)
```

Other aspects explored include similarity in terms of duration such as in `session_240922_003003`:

```
a = MIRLCaProxy.new
a.diss(0.12345)
a.whatdur // 26.2020
a.dur(26.2020)
```

4.2.3 Exploration of Randomness Through Number Series

The use of MIRLCa can involve randomness turned into serendipity (Xambó 2023). Apart from using the method `.random` (which is one of the functions that is unpredictable by our model), the first author has also investigated other ways of bringing randomness to the musical process. This includes using number series in different ways, such as the Fibonacci sequence (0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...) applied in different ways in `session_240918_232131`:

```
a = MIRLCaProxy.new
a.bpm(0)
a.bpm(1)
a.bpm(1)
a.bpm(2)
a.bpm(3)
a.bpm(5)
a.bpm(8)
a.bpm(13)
a.bpm(21)
a.bpm(34)
a.bpm(55)
a.bpm(89)
a.bpm(144)
```

In another session, the live coder explored the concept of rolling a dice to decide which of of the eight methods to write employing a random function (`session_240928_203633`). The use of other number series applied to the values of the parameters has been also explored in `session_240929_110801` to retrieve the sounds uploaded to Freesound using arithmetic series or to consult what sounds can result from applying small values to pitch, duration and bpm using geometric series in `session_240929_112115`.

4.2.4 The Role of Repetition

Repetition in music is a recurring motif that helps shape the diversity of sounds, often guiding them toward a more minimalistic musical result. It can also be used as a technique to overcome the lack of enough candidate sounds from a given query in MIRLCa. For example, `session_240925_000825` explores the same tags. While `"interferences"` gives a wide variety of sonic results under the same musical scope, `"interference"` retrieves the same sound repeatedly, starting at different moments each time, thereby creating potential rhythmic delay effects.

```
a = MIRLCaProxy.new
a.tag("interferences")
a.similar(0)
a.tag("interference")
a.similar(3)
a.tag("interference")
a.similar(5)
b = MIRLCaProxy.new
b.tag("interferences")
b.similar(0)
b.tag("interference")
b.similar(3)
b.tag("interference")
b.similar(5)
```

5 Discussion

We revisit here our research question: *What can building a dataset of your personal live coding style tell us about our personal style of live coding?*

In the process of building a dataset of a personal live coding style using the custom-made MIRLCaProxy library, we have found that it is essential to favour the biases to favour the personal style. Data bias is of big issue in artificial intelligence systems to avoid the promotion of inequality biases such as racism or sexism (D'Ignazio and Klein 2023; Jourdan and Caramiaux 2023). In artistic applications, the need to keep personal bias was highlighted by Murray-Browne and Tigas (Murray-Browne and Tigas 2021) acknowledging that small datasets are commonly generated by individuals for particular artistic work and their deliberate biases are often a signature of the musical piece.

A limitation of our approach is that it differs from the concept of artists manipulating small and quick datasets to become a larger-scale training process that requires some days of training. Turning the training into a rehearsal space of sonic explorations can motivate the artist to keep recording sessions, but the authors acknowledge that this approach is labour-demanding and can become unsustainable. This could be enhanced with automation that could be provided by deep learning, tools that can help generate similar code from existing material or bringing other like-minded live coders to contribute to the recordings. The training process with 800 data points marks the beginning of a journey that has already revealed new creative avenues, while the dataset itself remains a work in progress and continues to grow at the time of this writing.

6 Conclusion

This paper presented the technical and creative process of building a dataset of a personal live coding style using MIRLCaProxy, a custom-made SuperCollider class built on top of the MIRLCa extension. This paper has reflected on how the training process becomes a rehearsal space that supports the development of a personal style through constraints. It also explored the role of biases in this context.

Future work includes improving the classification model and adding a regression model to predict the value of the first argument of the suggested method. We also plan to integrate this code into the MIRLCa class as part of the features of the virtual agent. An open question is how the prediction will work with fully-featured live coding sessions (instead of narrowing it down to eight methods) using other functions available in MIRLCa such as automatic functions and sound effects. The possibility of voting the sonic passages (for example, the most successful ones) could help refine the learning of the personal style. We are also interested in evaluating the qualitative value of our approach for the human live coder. We acknowledge that as future work we should try other models such as Markovian, large language models,

or recurrent models, which would suit well for predicting sequences of actions and might offer comparative insight or potentially improved performance.

Altogether, this research demonstrates how machine learning can be integrated into live coding to learn from a personal coding style based on algorithmic digital sampling of crowd-sourced sounds, where biases may serve as creative assets.

6.1 Acknowledgments

The MIRLCa project was funded by the EPSRC HDI Network Plus Grant (EP/R045178/1).

References

Bernardo, Francisco, Chris Kiefer, and Thor Magnusson. 2020. "A Signal Engine for a Live Coding Language Ecosystem." *Journal of the Audio Engineering Society* 68 (10): 756–66.

Collins, Nick. 2011. "LL: Listening and Learning in an Interactive Improvisation System."

D'Ignazio, Catherine, and Lauren F Klein. 2023. *Data Feminism.* MIT Press.

Fails, Jerry Alan, and Dan R Olsen Jr. 2003. "Interactive Machine Learning." In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, 39–45.

Fiebrink, Rebecca, and Baptiste Caramiaux. 2018. "The Machine Learning Algorithm as Creative Musical Tool." In *The Oxford Handbook of Algorithmic Music*, edited by McLean Dean R. T., 181–208. Oxford, United Kingdom: Oxford University Press.

Font, Frederic, Gerard Roma, and Xavier Serra. 2013. "Freesound Technical Demo." In *Proceedings of the 21st ACM International Conference on Multimedia*, 411–12.

Jourdan, Théo, and Baptiste Caramiaux. 2023. "Machine Learning for Musical Expression: A Systematic Literature Review." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 319–31.

Landy, Leigh. 2007. *Understanding the Art of Sound Organization.* MIT Press.

Magnusson, Thor. 2010. "Designing Constraints: Composing and Performing with Digital Musical Systems." *Computer Music Journal* 34 (4): 62–73.

Murray-Browne, Tim, and Panagiotis Tigas. 2021. "Latent Mappings: Generating Open-Ended Expressive Mappings Using Variational Autoencoders." In *Proceedings of the International Conference on New Interfaces for Musical Expression.* Shanghai, China.

Reppel, Niklas. 2020. "The mégra System-Small Data Music Composition and Live Coding Performance." In *Proceedings of the 2020 International Conference on Live Coding*, 95–104.

Tanimoto, Steven L. 2013. "A Perspective on the Evolution of Live Programming." In *2013 1st International Workshop on Live Programming (LIVE)*, 31–34. IEEE.

Tremblay, Pierre Alexandre, Gerard Roma, and Owen Green. 2021. "Enabling Programmatic Data Mining as Musicking: The Fluid Corpus Manipulation Toolkit." *Computer Music Journal* 45 (2): 9–23.

Xambó, Anna. 2023. "Discovering Creative Commons Sounds in Live Coding." *Organised Sound* 28 (2): 276–89.

Xambó, Anna, Alexander Lerch, and Jason Freeman. 2018. "Music Information Retrieval in Live Coding: A Theoretical Framework." *Computer Music Journal* 42 (4): 9–25.

Xambó, Anna, Gerard Roma, Sam Roig, and Eduard Solaz. 2021. "Live Coding with the Cloud and a Virtual Agent." In *Proceedings of the New Interfaces for Musical Expression.*