Revista Vórtex | Vortex Music Journal | ISSN 2317–9937 | http://vortex.unespar.edu.br/
D.O.I.: https://doi.org/10.33871/23179937.2022.10.1.4695

A Networked Hybrid Interface for Audience Sonification and Machine Learning

Konstantinos Vasilakos

Istanbul Technical University, Center for Advanced Studies in Music | Turkey

Abstract: Lick the Toad is an ongoing project developed as a web based interface that runs in modern browsers. It provides a custom made platform to collect user data accessed from mobile devices, such as smartphones, tablets etc. The system offers a tool for interactive collective sonification supporting networked music performance. It can be used in various contexts, such as an onsite installation, or for the distribution of raw data for live coding performances making it a versatile component for an array of creative practices. Of these, live coding which is one of the author's artistic approach to create live performances is demonstrated in this article highlighting and elaborating on technical and musical aspects of this approach. Final sections outline the system as a tool for live coding performances and cover a series of potential interactions integrating audience and/or using it independently alike.

Keywords: Sonifications, Live Coding, Networked Music System, Improvisation, Web Application.

Received on: September 29th, 2021. Approved on: October 27th, 2021. Online available on: April 30th, 2022. Presented at the 18th Brazilian Symposium on Computer Music (2021). Guest editors: Dr. Ivan Eiji Simurra (UFAC), Dr. Flávio Schiavoni (UFSJ), Dr. Cristiano Figueiró (UFBA), Dra. Isabel Nogueira (UFRGS) e Dr. Tiago Tavares (Unicamp). Creative Commons CC-BY-NC.

The idea of network as a means for artistic experimentation is not anew in the field of electronic music. It goes back to the pioneering Telharmonium by an American inventor Thaddeus Cahill (MANNING, 2004, p.157) who was also influenced by others, such as Soemmerring's Musical Telegraph, and Tivadar Puskás' Telefonhírmondó, and the Musical Telegraph by Elisha Grey (CRAB, 2013). What was common in all these, was that they all attempted to enact the transmission of sound generated by custom made electric instruments via hitherto available broadcasting technology e.g., telegraphy and landlines. One can argue that these can be seen as primitive attempts for music distribution deploying network, which presumably paved the way to a new dimension of musical practice and artistic expression, as well as opening new challenges for artists. It allowed for the decentralization of the sonic information by transmitting the performance to various ends using radio and wired networks for telecommunications. One can see an arguably far fetched analogy with much of the current practices in the broad landscape of Sonic Arts, such as live coding and laptop ensembles which are using ad hoc networks as a means to establish real time communication and distribution of various data amongst each other during their improvisation, such as the Hub's initial attempts (GRESHAM-LANCASTER, 1998) for this paradigm of computer music networked performance. Of these, the Powerbooks Unplugged[1] and the Birmingham Ensemble for Electroacoustic Research (BEER)[2] are using custom made libraries for network communication, such as, the Republic[3] and Utopia[4] respectively. Both running in SuperCollider[5] programming language as external libraries, and are used to provide coders an interaction platform in order to communicate with each other as a means to thrust their live coding performances, making ad hoc networks inherent part of their research and musical practice.

In the wake of this performance paradigm, it helped to address some questions about *telepresence* and *liveness* via remote and distributed performers using network music systems and remote communication. Besides the distribution of various data types through ad-hoc local network some other examples using audio signals via the internet enacting remote communication include BEER's "Recalibrated" performance of a live coding piece entitled *Pea Stew*, an adaptation of Nick

[1] See http://wertlos.org/pbup/
[2] See http://www.beast.bham.ac.uk/offspring/beer/
[3] https://github.com/musikinformatik/Republic
[4] https://github.com/muellmusik/Utopia
[5] https://supercollider.github.io/

Collins' Pea Soup, and the Symphony in Blue 2.0 by the Istanbul Coding Ensemble[6], a live coding performance based in Kamran Ince's composition entitled Symphony in Blue (2012)[7]. While the former piece involves the feedback of audio signals amongst remote coders the latter uses piano signal for machine listening utilization both locally and remotely using SonoBus[8], a low latency state of the art audio transmission software. Lastly, besides network being used as a medium to distribute audio information and/or data, some works have used directly its inherent elements, for example the Network Ensemble has used real time sound representations of raw packets from WiFi networks (SMITH AND TACCHINI, 2017), and Chris Chafe's music work entitled Ping (CHAFE, 2001) used the network's ping latency information to modulate sound synthesis parameters.

1. Sonification and auditory display

Sonification is the practice of rendering data to sound (HERMANN et al., 2011; WORRALL, 2019). Real time streams of data or static data sets can be used as raw material to build wave forms or create associations between data streams and sound synthesis parameters. This simple yet fertile approach for experimentation has been widely explored in the field of the Sonic Arts and digital arts alike. Examples include and the deployment of astronomic and environmental readings to particle collisions data.

In addition, sonification practice has been the workhorse for both artistic and educational outreach. Given the technical affordances sonification can be explored as web applications and can run in browsers of mobile devices, such as smartphones, tablets, and whatnot. This provides a personal exploration tool and at the same time it offers decentralized environments[9] which were previously unexplored due to the need of specialized background. Consequently, this opens up to non-technical backgrounds and the interested layman. Similarities in approach, includes IPSOS[10] (VASILAKOS et al., 2020), a web-based application that runs on any modern browser and allows for the real-time sonification of data from the Large Hadron Collider, at CERN, in Switzerland.

[6] See https://konvas.netlify.app/ice/
[7] See http://www.kamranince.com/html/compositions.php?id=189&page=
[8] https://sonobus.net
[9] Decentralized environments refer to the standard model of laptop performances where the interaction relies on the standard human computer relationship with the sonic generation.
[10] http://ipsos.web.cern.ch/

2. Machine learning and the (Sonic) arts

Some examples that highlight using machine learning in music practice include Marije Baalman's "The Machine is Learning" (BAALMAN, 2020), Sonami's and Fiebrink's (FIEBRINK AND SONAMI, 2020), and Sturm's work in the field (L. Sturm and Ben-Tal, 2017). Due to the development of machine learning modules in the browser, it allows for control without the need for specialized skills and interact with training and control of neural networks. This seems to be also observed by Amershi: "IML enables even users with little or no machine-learning expertise [to] steer machine-learning behaviors through low-cost trial and error or focused experimentation with inputs and outputs" (AMERSHI et al., 2014).

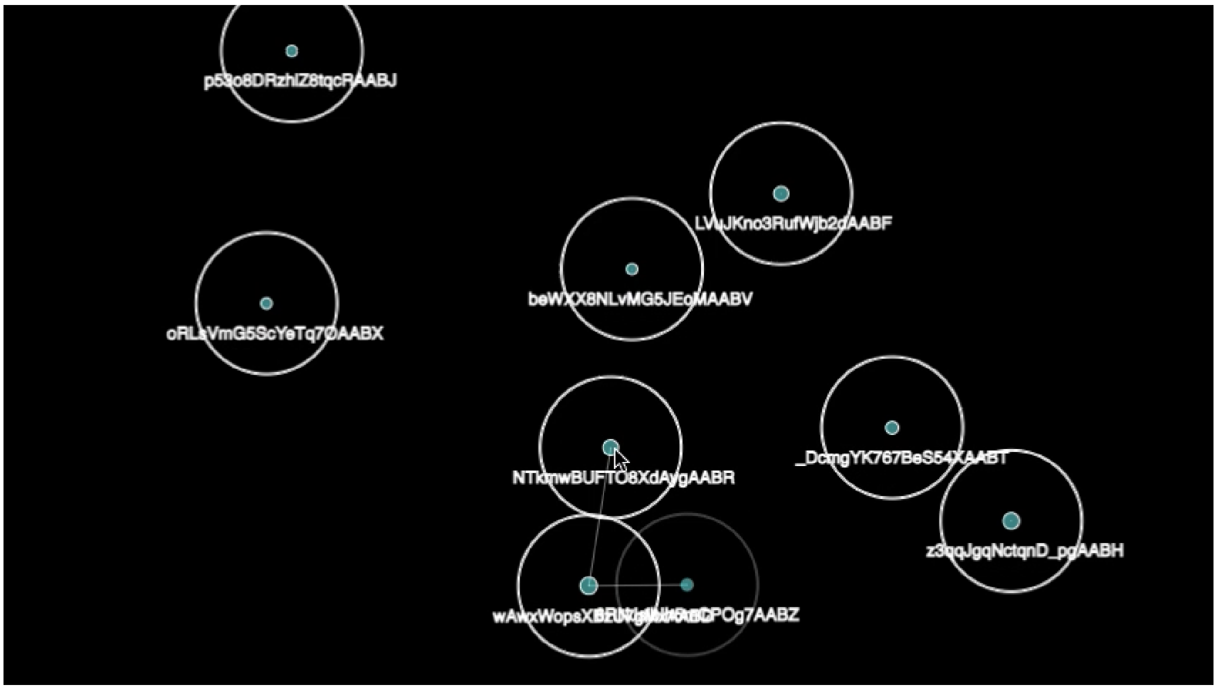
3. Lick the Toad

Lick the Toad offers an interface to interact with other users that are connected in an ad-hoc network and share their positions to one another and at the same time train a model of a neural network using this activity. The interface runs in mobile devices and modern web browsers as seen in figure 1.

The neural network is implemented using the ML5 library[11], which offers monitoring possibilities as seen below in figure 2. For an overview of the system see this link[12].

[11] https://learn.ml5js.org/#/reference/neural-network
[12] https://tinyurl.com/2jk5bvyf

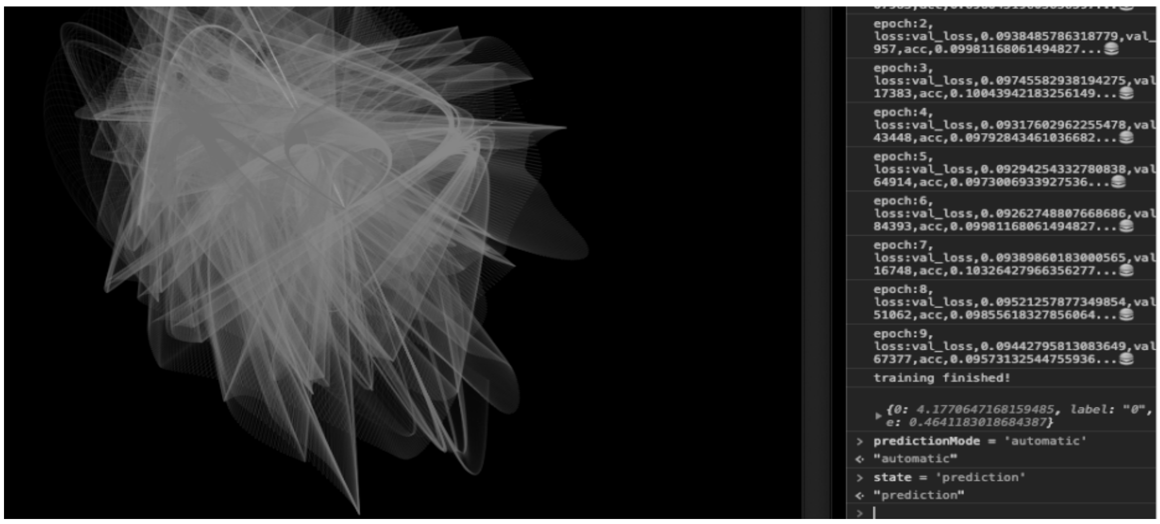
Figure 1 – User Interface of Lick The Toad.



It is an open source project and can be found at this link[13]. Besides of the training data generation the users can also engage in collective prediction process and collaborate with each other to build a collective work e.g., providing it is placed as an interactive sound installation. Due to the variability of the data, it can be safely speculated that it may allow for a wide range of unpredictability of the musical outcome. This assumes the unlimited associations between data streams and sound, enabled by mappings that one has to establish with the sound generation modules in advance. The project is using SuperCollider but any platform such as Max/MSP[14] and Pure Data[15] can be employed using Open Sound Control[16] (OSC) communication. Thus, the platform allows for active engagement with the sound generation instead of experiencing a musical work that was created with a digital musical instrument (e.g., as seen broadly in NIME community), which often may be viewed by the audience as a black box requiring programming dexterity and/or technical background in music technology.

[13] https://github.com/KonVas/lick-the-toad
[14] https://cycling74.com
[15] https://puredata.info
[16] https://www.cnmat.berkeley.edu/opensoundcontrol

Figure 2 – Visualization of Collected Values and Training.



```
epoch:2,
loss:val_loss,0.0938485786318779,val_
957,acc,0.09981168061494827...
epoch:3,
loss:val_loss,0.09745582938194275,val
17383,acc,0.10043942183256149...
epoch:4,
loss:val_loss,0.09317602962255478,val
43448,acc,0.09792843461036682...
epoch:5,
loss:val_loss,0.09294254332780838,val
64914,acc,0.0973006933927536...
epoch:6,
loss:val_loss,0.09262748807668686,val
84393,acc,0.09981168061494827...
epoch:7,
loss:val_loss,0.09389860183000565,val
16748,acc,0.10326427966356277...
epoch:8,
loss:val_loss,0.09521257877349854,val
51062,acc,0.09855618327856064...
epoch:9,
loss:val_loss,0.09442795813083649,val
67377,acc,0.09573132544755936...
training finished!

 ▸{0: 4.1770647168159485, label: "0",
  e: 0.4641183018684387}
> predictionMode = 'automatic'
<· "automatic"
> state = 'prediction'
<· "prediction"
> |
```

3.1 Collection and training

The interface involves a single object which transmits to all connected users continuously its *X* and *Y* positions as float values relative to the screen borders of the device, see figure 1.

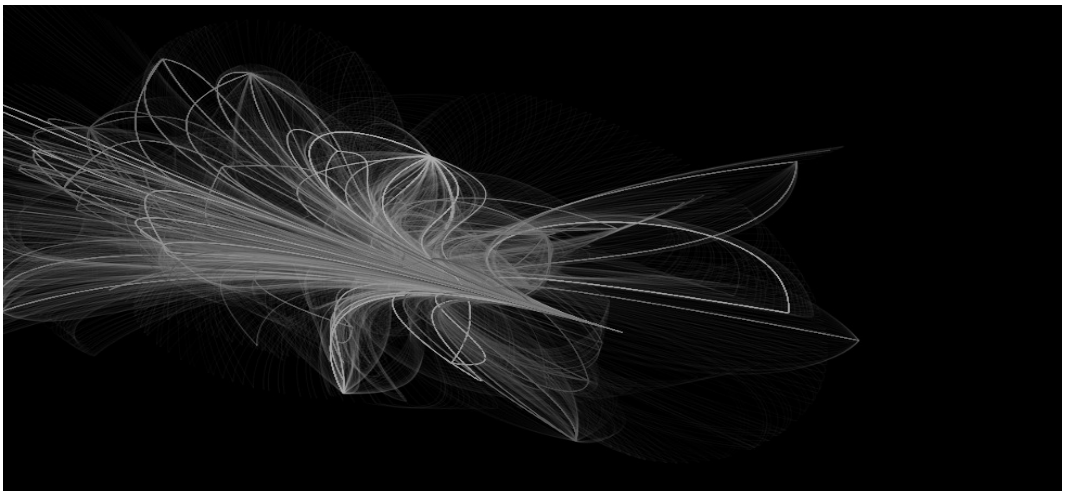
While someone may interact with the rest of the connected users they can see a string of the ID, which represents each user currently in the system. The system then is able to collect the data and render it to a normalized range of 0.0 – 1.0 and start the training process.

In addition to the *X* and *Y* values of the object, the user is able to add a target/output value. A snippet of the collected data prior of normalization can be seen in the "*data*" JSON snippet at the end of this paragraph. The *YS* is the output value, that is the targeted value for prediction here represented as *frequency*. Numerous iterations and similar data sets using this template are stored in JSON format, the file can be saved locally in order to load the model and/or run the training process again later. One can add other targets instead of *frequency* according to the needs providing that it follows the same format. The interface also creates visualizations of the receiving data using Bezier curves [17] as seen in figure 3.

[17] https://tinyurl.com/ykd6ywnu

```
{
  "data": [
    {
      "xs": {
        "x": 226.322645526962,
        "y": 84.91814510458188
      },
      "ys": {
        "frequency": 660
      }
    }
]
}
```

Figure 3 – Bezier Incoming Data.



3.2 Training the model

The platform offers a minimal interface to start the various actions of the interface, such as collecting and training the model of the neural network. These actions are triggered using the native keyboard of the device.

3.3 General overview of the interface.

It can be also monitored via the terminal window of the web page providing information about *loss, epochs* etc. of the training progress (see figure 2).

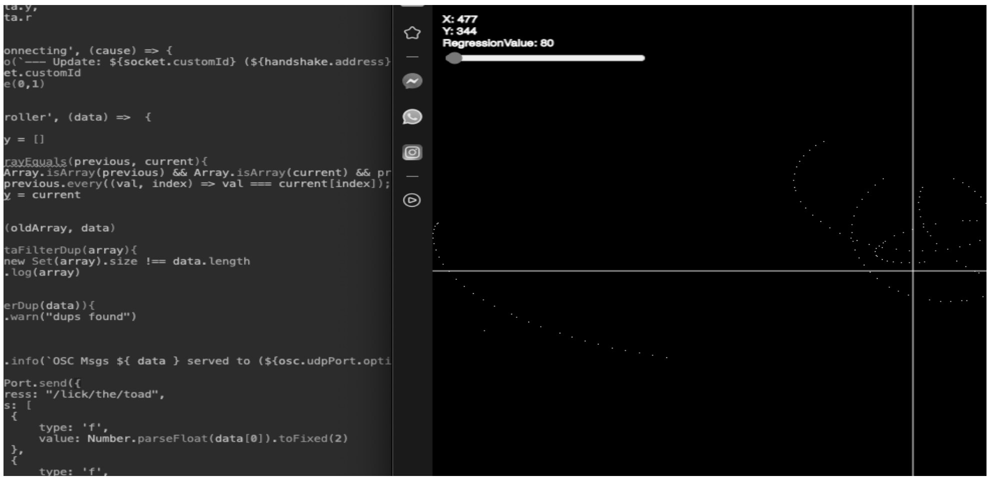
3.4 Prediction and output

Once the system completes the training it starts generating prediction values based on the model, and triggering float values corresponding between the proximal points, for example, a user might enter some input data and declare it as *low*, *mid*, and *high*, their corresponding values (80, 220, 660) respectively; The output value will provide a continuous float number known as *regression value* (=x) between the targets, that is between *low* and *mid*, then this will be:

$$low \leq x \leq med$$

The selection is based on a cursor according to its relative *X, Y* positions. This is controlled manually or randomly using a Perlin noise operator [18] as shown in figure 4.

Figure 4 – Cursors Controls Trigerring Regression Data.



```
ta.y,
ta.r

onnecting', (cause) => {
o(`--- Update: ${socket.customId} (${handshake.address}
et.customId
e(0,1)

roller', (data) =>  {

y = []

rayEquals(previous, current){
Array.isArray(previous) && Array.isArray(current) && pr
previous.every((val, index) => val === current[index]);
y = current

(oldArray, data)

taFilterDup(array){
new Set(array).size !== data.length
.log(array)

erDup(data)){
.warn("dups found")

.info(`OSC Msgs ${ data } served to (${osc.udpPort.opti

Port.send({
ress: "/lick/the/toad",
s: [
 {
     type: 'f',
     value: Number.parseFloat(data[0]).toFixed(2)
 },
 {
     type: 'f',
```

```
X: 477
Y: 344
RegressionValue: 80
```

3.5 Mapping interaction

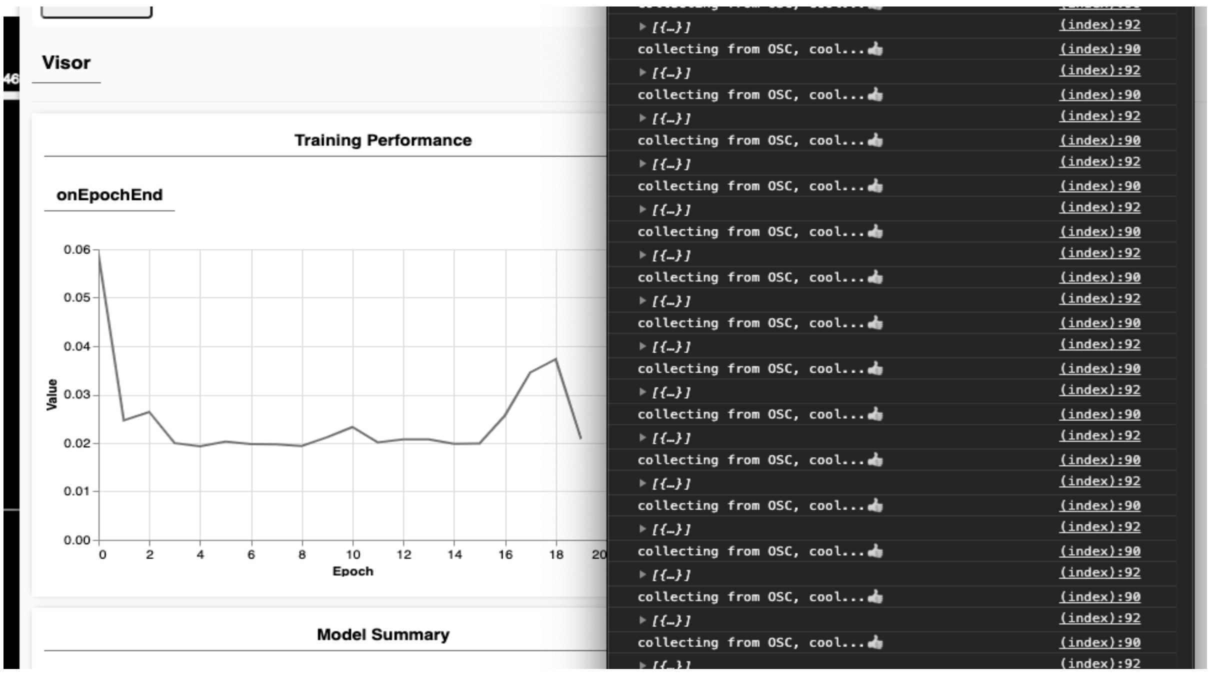
In addition to sending the regression value for sound mapping, the system also outputs the *X* and *Y* position of the cursor, that is, *X* and *Y* float numbers. For example, providing the cursor is between two points that are logged as *low* (=80.0) and *mid* (=220.0) the system will provide a value

[18] https://p5js.org/reference/#/p5/noise

according to the proximity of these two and the same stream of the OSC message *X* and *Y* will be added. This format was found to be the most convenient for mapping the values to higher level musical parameters, such as frequency or duration to synthesize a sonic event.

The system can host diverse classification according to the nature of the input/output data, and thus generate a broad range of value range and mapping with sound sources accordingly. That allows the system to provide a versatile tool for experimentation in a sound synthesis context, instead of something that can be used only in a specific context and/or predetermined way which was devised by the author. For example, in live coding performances the system can acquire other data types to train the neural network. Successful tests using data to train the model generated from SuperCollider and used for live coding are also possible as seen in figure 5.

Figure 5 – Collecting from OSC by SuperCollider.



```
▶ [{…}]                                      (index):92
collecting from OSC, cool...👍               (index):90
▶ [{…}]                                      (index):92
collecting from OSC, cool...👍               (index):90
▶ [{…}]                                      (index):92
collecting from OSC, cool...👍               (index):90
▶ [{…}]                                      (index):92
collecting from OSC, cool...👍               (index):90
▶ [{…}]                                      (index):92
collecting from OSC, cool...👍               (index):90
▶ [{…}]                                      (index):92
collecting from OSC, cool...👍               (index):90
▶ [{…}]                                      (index):92
collecting from OSC, cool...👍               (index):90
▶ [{…}]                                      (index):92
collecting from OSC, cool...👍               (index):90
▶ [{…}]                                      (index):92
collecting from OSC, cool...👍               (index):90
▶ [{…}]                                      (index):92
collecting from OSC, cool...👍               (index):90
▶ [{…}]                                      (index):92
collecting from OSC, cool...👍               (index):90
▶ [{…}]                                      (index):92
collecting from OSC, cool...👍               (index):90
▶ [{…}]                                      (index):92
collecting from OSC, cool...👍               (index):90
▶ [{…}]                                      (index):92
collecting from OSC, cool...👍               (index):90
▶ [{…}]                                      (index):92
```

After the model is created it can be stored locally in the computer's hard disk and load again later for using it with other sound sources and software for musical creation.

3.6 Technical Details

TensorFlowJS[19], is a library for machine learning in the browser. ML5 is built on top of TensorFlow. It offers a higher level interface to implement machine learning in the browser including neural networks, pose recognition, and image prediction using pre-trained models offered by MobileNet[20]. More information about the library is found at this link[21]. To provide real time data distribution amongst users the project is using Socket.io[22], a library that is designed to provide to applications real time communications, such as event handling. To communicate with SuperCollider the project uses OSC communication implemented with OSC.js[23] library. NodeJS[24], the runtime of JavaScript also provides a local web server, which is built on top of the interface. This is also tested on a Raspberry Pi 3 Model B[25]. The visualizations are built in P5JS[26], a link to the script can be found at this link[27].

4. Creative Outcomes

While the project is designed to support a diverse context of sound making and neural network integration, a selection of potential settings is covered in order to demonstrate some of the capabilities of the system, that is, live coding at its initial stages of explorations with Lick The Toad.

More specifically, the system is a non-opinionated application that can be used in various ways such as, using data from logged users as onsite installation using data from bystanders, or various streams generated by other applications i.e., SuperCollider, and thus it can be presumed that it may be an interesting asset in an array of creative settings as follows.

[19] https://www.tensorflow.org/js
[20] https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet
[21] https://ml5js.org/
[22] https://socket.io
[23] https://github.com/colinbdclark/osc.js
[24] https://nodejs.org/en/
[25] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[26] https://p5js.org/
[27] https://github.com/KonVas/lick-the-toad

Next sections outline the system as a tool for live coding performances and cover a series of potential interactions integrating audience and/or using it independently alike with SuperCollider.

4.1 Live coding and audience participation

In the context of music, live coding is the real time alteration of source code of programs that generate sound or visuals. A performer is able to interfere in the sound synthesis processes by modifying the state of the program while it is running by adding, suspending and changing the structure of the algorithms at hand (COLLINS et al., 2003; NILSON, 2007; ZMÖLNIG and ECKEL, 2007) (COLLINS et al., 2003; NILSON, 2007; ZMÖLNIG and ECKEL, 2007).

It has been established as an emerging computer music paradigm with many enthusiasts using various tools and environments for improvising and tweaking source code in front of the audience, as well as forming various communities, such as the Algorave (LUKA, 2019), and laptop ensembles, such as BEER, Cybernetic Orchestra[28], and PowerBooks Unplugged.

Live coding community in general has a dedicated organization named Toplap[29], the home of live coding, which also includes a list with notable artists as well as information about tools and environments or events dedicated in this performance practice.

Audience participation has been deployed in computer music practice in many forms using various tools and media empowered by web technologies and commonly used hardware i.e., smartphones and tablets (ARAÚJO et al., 2019; MATUSZEWSKI, SCHNELL and BEVILACQUA, 2019; HÖDL et al., 2020; VASILAKOS et al., 2020) to build collective interactions amongst local and remote users alike.

Lick the Toad aims to provide a general platform to interact with the audience and communicate while live coding. That is, the audience is able to steer the coder's decision by sending data that is generated from their smartphones. Turning live coding performances look less esoteric and opening to the public not only in a conceptual level but in a collaborative way.

[28] https://global.mcmaster.ca/activity/cybernetic-orchestra/
[29] https://toplap.org/wiki/Main_Page

This enables them to engage in the performance instead of behaving passively as sole auditors during the performance. While this opens up to new possibilities for live music making, it also poses new challenges, which can be examined using this platform embedding neural networked capabilities while exploring with live coding improvisation techniques.

Specifically, one of the practical uses of the system is that it can reflect on the questions which arises by such interaction, that is, from live dialogue between the audience and coder, and how the former is influencing the later's decision while performing.

The following elaborates in a demonstration of a video[30] which was created as a proof of concept. It aims to demonstrate the system as a pilot strategy as in real life circumstances i.e., used with more than one user input from audience.

While live coding is an improvised performance practice by definition, it can embody other dynamic inputs, such as indeterministic data ranges which arise by the incoming streams of data of the system. For example, providing decisions in the mapping between the incoming data and sound synthesis parameters.

Incoming data from the platform is received from the system using OSC including the regression values and x, y, values of the cursor from the user's interface (see figure 3). The OSC receivers can be adjusted and modified in real time without breaking the flow of the streaming input.

Another potential approach the performance would include, generating the prediction values and streaming them to SuperCollider, which it can adapt the data in additional functions that entail specific sounds to be triggered alike, e.g., *low, mid, high*, yet a very limited example for expressing the concept, as the targets can be adapted to much more detail and provide more sophisticated predictions based on the trained model.

The coder then is also able to improvise with the sound generation sources in SuperCollider in real time using all live coding techniques accordingly, (e.g., using JITLib in SC). During this performance one can explore live mapping strategies and its variants between the classification/prediction streams from the platform and the sound design algorithms generating an unpredictable musical outcome every time the system reiterates over the generation of prediction data.

[30] Find the video at this link (https://www.youtube.com/watch?v=c6P8fg9f8qw).

The regression values are written in a buffer which is rendered in a format which a custom made player is reproducing the contents of the buffer. Therefore, the performance in the video elaborates on using this as audio source for several post processing modules, such as pitch shifting, and tweaking its parameters using live coding techniques while it is running, as seen at end of the video.

Any mapping is possible but for demonstration purposes the brief video shows incoming regression values being mapped to several parameters of a sounding module. At the first seconds of the video the streams are viewed in the bottom of the SuperCollider application's, which is known as the post window.

At the same time a window of graphic interface with faders shows the controls of the sounding modules fluctuating due to the variability of the regression values.

The interaction scenario and goal of the prediction of the neural net here is that some input values were fed and classified as low, med, and high. That means that when the cursor which represents user's movements in the interface will associate the x, and y values of the cursor with the targeted values that were used during the training. Therefore, the regression values correspond to these values accordingly. But as mentioned already any target values can be used and adjust the system with minimum tweaking of the source code of the platform to provide functionality for diverse inputs and targeting (output) values accordingly.

Further more in the video demonstration the live coding performance shows how the coder is modifying the source code of the sound algorithms and the parameters of those in real time, allowing for a wider range of sound manipulation and musical interpretation of the incoming streams of values.

4.1.1 Training with SuperCollider

While the idea of training started initially by using user's data when designing the system, for the live coding performances the input can be generated from SuperCollider alike. That is, any sort of data that can used from SuperCollider's data generators or an external interface, such a hardware controllers bind to SuperCollider, can be used to train the neural network. In this way, the live performance can take place more autonomously, for example, the coder will be generating some input and train the system. For demonstration purposes, some random operators were used to generate some values that are associated with the targeted labels, that is, low, mid, high. Code below shows the implementation. Streaming data was communicated to the server of the app using OSC from SuperCollider (see figure 5 and figure 6).

```
(
n = NetAddr("127.0.0.1", 57121); // To Server's port number.
~setter = {
|state = 0|
Tdef(\oscToJS, {
loop{
0.25.wait;
if(state == 1){
//These are sending the messages to bind target values for training.
n.sendMsg('/osc-to-browser', state, "high", rrand(10.0, 1439.0), rrand(0.1,
230.0));//low range
n.sendMsg('/osc-to-browser', state, "mid", rrand(10.0, 1439.0), rrand(230,
430.0));//mid range
n.sendMsg('/osc-to-browser', state, "low", rrand(10.0, 1439.0), rrand(430.0,
726.0));//high range
"% state was changed to trigger %".format(state).postln;
} {
n.sendMsg('/osc-to-browser', " ");
}
}
});
}
)
```

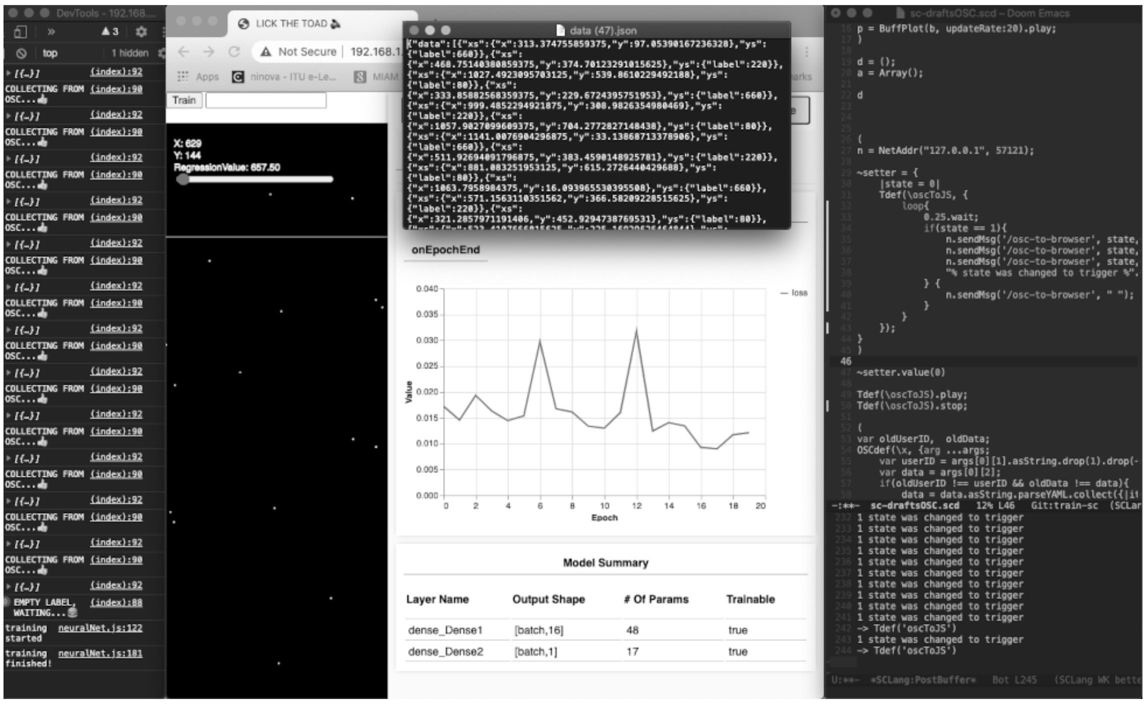
The figure (6) shows the code in SuperCollider generating data in order to create the input values and targets alike and also illustrated in the above chunk of code and attached to this link[31]. With this approach the coder can work independently and while the system can predict values automatically as mentioned already using a Perlin noise generator to fluctuate the cursor then the system can behave as an autonomous agent while coding. This could also use some operators to control the cursor of the system generated by SuperCollider as well allowing to use the system in various contexts.

Therefore, in live coding settings the data can be mapped to any available parameters which can be developed live or prior to performance and thus the performance strategy can focus on the efficiency of mapping strategies and improvise with various aspects of those such as, range specifications and their alteration dynamically.

[31] https://github.com/KonVas/lick-the-toad/blob/main/SuperCollider/RecordDataToBuffer.scd

That is, the implementation would control parameters using the incoming streams (x,y, regression values) in order to create an indeterministic behavior which could be moderated using live coding techniques and real time balancing of range specifications and other techniques applicable to control data and sound synthesis parameters. For example, many to many or one to many bindings.

Figure 6 – Collection of Data from SuperCollider and Training of Lick The Toad.



```
▸ [{…}]              (index):92
COLLECTING FROM      (index):90
OSC...👍
▸ [{…}]              (index):92
COLLECTING FROM      (index):90
OSC...👍
▸ [{…}]              (index):92
COLLECTING FROM      (index):90
OSC...👍
▸ [{…}]              (index):92
COLLECTING FROM      (index):90
OSC...👍
▸ [{…}]              (index):92
COLLECTING FROM      (index):90
OSC...👍
▸ [{…}]              (index):92
COLLECTING FROM      (index):90
OSC...👍
▸ [{…}]              (index):92
COLLECTING FROM      (index):90
OSC...👍
▸ [{…}]              (index):92
COLLECTING FROM      (index):90
OSC...👍
▸ [{…}]              (index):92
COLLECTING FROM      (index):90
OSC...👍
▸ [{…}]              (index):92
COLLECTING FROM      (index):90
OSC...👍
▸ [{…}]              (index):92
COLLECTING FROM      (index):90
OSC...👍
▸ [{…}]              (index):92
COLLECTING FROM      (index):90
OSC...👍
▸ [{…}]              (index):92
COLLECTING FROM      (index):90
OSC...👍
▸ [{…}]              (index):92
EMPTY LABEL,         (index):88
WAITING...
training        neuralNet.js:122
started
training        neuralNet.js:101
finished!
```

{"data":[{"xs":{"x":313.374755859375,"y":97.05390167236328},"ys":{"label":660}},{"xs":{"x":468.75140380859375,"y":374.70123291015625},"ys":{"label":220}},{"xs":{"x":1027.4923095703125,"y":539.8610229492188},"ys":{"label":80}},{"xs":{"x":333.85882568359375,"y":229.6724395751953},"ys":{"label":660}},{"xs":{"x":999.4852294921875,"y":308.9826354980469},"ys":{"label":220}},{"xs":{"x":1057.9027099609375,"y":704.2772827148438},"ys":{"label":80}},{"xs":{"x":1141.0076904296875,"y":33.13868713378906},"ys":{"label":660}},{"xs":{"x":511.92694091796875,"y":383.4590148925781},"ys":{"label":220}},{"xs":{"x":881.083251953125,"y":615.2726440429688},"ys":{"label":80}},{"xs":{"x":1063.7958984375,"y":16.093965530395508},"ys":{"label":660}},{"xs":{"x":571.1563110351562,"y":366.58209228515625},"ys":{"label":220}},{"xs":{"x":321.2857971191406,"y":452.9294738769531},"ys":{"label":80}},

Layer Name	Output Shape	# Of Params	Trainable
dense_Dense1	[batch,16]	48	true
dense_Dense2	[batch,1]	17	true

```
p = BuffPlot(b, updateRate:20).play;
)

d = ();
a = Array();

d



(
n = NetAddr("127.0.0.1", 57121);

~setter = {
    |state = 0|
    Tdef(\oscToJS, {
        loop{
            0.25.wait;
            if(state == 1){
                n.sendMsg('/osc-to-browser', state,
                n.sendMsg('/osc-to-browser', state,
                n.sendMsg('/osc-to-browser', state,
                "% state was changed to trigger %".
            } {
                n.sendMsg('/osc-to-browser', " ");
            }
        }
    });
}
)

~setter.value(0)

Tdef(\oscToJS).play;
Tdef(\oscToJS).stop;

(
var oldUserID,  oldData;
OSCdef(\x, {arg ...args;
    var userID = args[0][1].asString.drop(1).drop(-
    var data = args[0][2];
    if(oldUserID !== userID && oldData !== data){
        data = data.asString.parseYAML.collect({|it
-:**-  sc-draftsOSC.scd   12% L46   Git:train-sc  (SCLar
1 state was changed to trigger
1 state was changed to trigger
1 state was changed to trigger
1 state was changed to trigger
1 state was changed to trigger
1 state was changed to trigger
1 state was changed to trigger
1 state was changed to trigger
1 state was changed to trigger
1 state was changed to trigger
-> Tdef('oscToJS')
1 state was changed to trigger
-> Tdef('oscToJS')
U:**-  *SCLang:PostBuffer*   Bot L245   (SCLang WK bette
```

4.2 As Onsite Autonomous Instrument

As mentioned briefly already, the system is able to provide a platform where users can use their smartphones to interact with each other and see others in the application controlling a host computer's sound design program. This can be used as onsite installation which will run with no live coding or other human intervention besides of controlling the data that fluctuates the sound design parameters. Future directions will include specific instructions regarding the installation process of the system and thus it will require other hardware and bespoke onsite hardware and loudspeakers specifications to host the system as an interactive audio-visual installation.

5. Future Directions

The application is at the stage of finalizing and running it via a local server in NodeJS. While it is an ongoing project, it is constantly adapted upon the requirements of projects. One of the future projects that the system may be used is with acoustic instrumentalists and train a model of the neural network using sensors for gestural information and interpreted in various ways.

The primary goal of the project is to make the system accessible using a mobile device for online interactions. That means the users can interact from distant locations to train the system as well as interpret the data to create telematic sonification work. One possible way to implement this reciprocal interaction between training and sonifying amongst remote user(s) is to embed a remote server project developed by Serkan Sevilgen, a demo of this project can be seen at this link[32].

It seems to provide a fertile solution to creating projects that focus on connecting remote or locally embodied interactions, in the case of remote settings, and the wake of isolation and remoteness periods due to the pandemic of COVID19 this may provide a useful tool for interaction and creative collaboration.

For this it is required to be deployed and hosted on a remote server; this remains to be implemented in due course and finalization of other features will be implemented accordingly. Thus, next steps include the deployment of the platform as an online app which can host remote interactions amongst users across the internet and distribute data for various purposes, e.g., on site sound synthesis clients or live coders alike. Finally, further directions of this project will provide the combination of cosmetic and more eloquently appealing visualizations of the data, both in training and prediction modes. Finally, another future implementation which can be considered to expand on the creative outcomes of the system will provide the interaction between sound events and their visualization in real time, that is the sound events to be create real time graphics in another page of the interface.

[32] https://www.youtube.com/watch?v=uP1_Jp-sfOc

6. Conclusion

With the advent of machine learning technologies in the browser and its distributed collaborations it opens a brave new area for experimentation and musical research. Placing the audience in the forefront of the compositional epicenter instead of being a passive listener enables the collective interaction and real time communication amongst users and performers alike. This will provide for another level of exploration of creative nuances but also challenges for computer music practitioners and digital artists. Lick the Toad provides an interface to collect data from mobile devices and train a model to create further real time sonifications, which can be both triggered by the participants and/or used for live coding performances alike. So far, the system runs as a local environment in ad-hoc network using NodeJS and SuperCollider for sonification purposes. The project has been proven to be quite flexible in terms of the collection of training data and thus it can be adapted to various projects, e.g., instruments and bystanders such as in the context of an on-site installation. Once the model is trained the interpretation can be mapped using OSC and SuperCollider as its sound engine. Further explorations will allow for a fully decentralization of the data collection and run as a mobile application hosted and deployed on a remote server. This will allow for a wider interaction amongst users across the network. The idea of using the audience as input during the performance will allow to open up a real time dialogue with the coder and influence their decisions.

REFERENCES

AMERSHI, S.; CAKMAK, M.; KNOX, W.B.; KULESZA, T. Power to the People: The Role of Humans in Interactive Machine Learning. *AI Magazine*, [online] 35(4), pp.105–120, 2014. https://doi.org/10.1609/aimag.v35i4.2513

BAALMAN, M. *the machine is learning*. [Personal Website] Marije Baalman. 2020. Available at: <https://marijebaalman.eu/projects/the-machine-is-learning.html> [Accessed 16 April 2021].

CHAFE, C. Ping. 2001.

COLLINS, N.; MCLEAN, A.; ROHRHUBER, J.; WARD, A. Live coding in laptop performance. *Organised Sound*, [online] 8(3), pp.321–330, 2003. https://doi.org/10.1017/S135577180300030X

CRAB, S. The 'Telharmonium' or 'Dynamophone' Thaddeus Cahill, USA 1897. *120 Years of Electronic Music*. 2013. Available at: <https://120years.net/wordpress/the-telharmonium-thaddeus-cahill-usa-1897/> [Accessed 5 April 2022].

FIEBRINK, R.; SONAMI, L. Reflections on Eight Years of Instrument Creation with Machine Learning. In: R. Michon and F. Schroeder, eds. *Proceedings of the International Conference on New Interfaces for Musical Expression*. [online] Birmingham, UK: Birmingham City University. pp.282–288, 2020. Available at: <pdfs/nime2020_paper45.pdf>

GRESHAM-LANCASTER, S. The Aesthetics and History of the Hub: The Effects of Changing Technology on Network Computer Music. Leonardo Music Journal 8 (1998): 39. https://doi.org/10.2307/1513398

HERMANN, T.; HUNT, A.; NEUHOFF, J. G. (eds). *The Sonification Handbook*. Berlin: Logos Verlag, 2011.

L. STURM, B.; BEN-TAL, O. Taking the Models back to Music Practice: Evaluating Generative Transcription Models built using Deep Learning.*Journal of Creative Music Systems*, [online] 2(1), 2017. https://doi.org/10.5920/JCMS.2017.09

LUKA, 2019. *Live Coding and Algorave*. [online] Medium. Available at: <https://lukabaramishvili.medium.com/live-coding-and-algorave-9eee0ca0217f> [Accessed 14 April 2021].

MANNING, P. *Electronic and computer music*. Rev. and expanded ed ed. Oxford ; New York: Oxford University Press, 2004.

NILSON, C. Live coding practice. In: *Proceedings of the 7th international conference on New interfaces for musical expression - NIME '07*. [online] the 7th international conference. New York, New York: ACM Press. p.112, 2007. https://doi.org/10.1145/1279740.1279760

SMITH, O.; TACCHINI, F. Network Ensemble Selected Network Studies. Number SF0010 in *The Strong of the Future*. Rizosfera, Via Guido de Ruggiero, 6 – 42123, Reggio Emilia – Italia, 2017.

VASILAKOS, K. Exploring Live Coding as Performance Practice. *2021*, [online] 21(Guz 2021), pp. 21–38, 2021. Available at: <https://dergipark.org.tr/tr/pub/porteakademik/issue/66105/1033868> [Accessed 7 February 2022].

VASILAKOS, K.; WILSON, S.; MCCAULEY, T.; YEUNG, T.W.; MARGETSON, E; KHOSRAVI MARDAKHEH, M. Sonification of High Energy Physics Data Using Live Coding and Web Based Interfaces. In: R. Michon and F. Schroeder, eds. *Proceedings of the International Conference on New Interfaces for Musical Expression*. [online] Birmingham, UK: Birmingham City University. pp. 464–470, 2020. Available at: <pdfs/nime2020_paper76.pdf>

WORRALL, D. *Sonification Design: From Data to Intelligible Soundfields*. Springer Publishing Company, Incorporated, 1st edition, 2019.

ZMÖLNIG, Io.; ECKEL, G. Live coding: An overview. *International Computer Music Conference, ICMC 2007*, pp. 295–298, 2007.

ABOUT THE AUTHOR

Konstantinos Vasilakos creates electro-acoustic music using hitherto available computer music systems within the milieu of interactive music and digital arts. He studied in Greece, UK and The Netherlands. His music interests include performance with hardware interfaces, live coding, sonification, and networked music environments for real time improvisation. ORCID: https://orcid.org/0000-0002-3058-0048. E-mail: konstantinos.vasilakos@gmail.com