

Planting and Shaping the Neural Tree: Growing a Musical Collaborator Through Objective-Free Neuroevolution in Open-Ended Exploration

Fabian Ostermann

Department of Computer Science
TU Dortmund University, Germany
fabian.ostermann@tu-dortmund.de

Tim Löhde

Visual Artist and Musician
Düsseldorf, Germany & Vienna, Austria
info@timloehde.de

Abstract

We present a live performance for experimental electronic music in which a continuously evolving neural network joins two human performers to form an improvising trio. Rather than training a generative model on pre-existing works, we use neuroevolution strategies to grow a feed-forward neural graph in real time, with no formal optimization objective. Following a tree metaphor, the performers engage the network in two complementary roles: one "plants" its seed by controlling parameter initialization and mutation, while the other "shapes" its canopy by mapping MIDI note and control events to different synthesizers and their parameters. Using the Godot game engine, the evolving topology is rendered visually as a live, wind-blown tree, giving the audience a direct view of the organism's growth. Built on the open-source BbMuse framework, our approach foregrounds lightweight, real-time AI that evolves alongside human musicians without the use of pre-curated data, treating the machine as a creative collaborator.

1 MOTIVATION

Current developments in AI music creation raise questions about the relationship between human and machine as creative collaborators. How can AI become a genuine musical partner for musicians rather than a replacement for them?

Our performance sets out to propose answers to this question, while raising new ones. Rather than creating music through huge pre-trained black-box models or neural networks trained on pre-existing non-original works, we want to enable a creative partnership, a living collaboration between musician and machine that emerges in the moment of the performance itself. Therefore, we developed the metaphor of a living organism and applied it to the theoretical concept of neural networks. Further, as we do not use a formally definable optimization objective, we decided to borrow concepts from neuroevolution to evolve the network topology and parameters over time, because we liked the idea of a growing tree-like construct. This allowed us to identify two inherently different kinds of interacting with the artificial organism:

- (A) **planting** the tree's seed: technically, this means controlling parameter initialization, topological growth, and mutation frequency
- (B) **shaping** the tree's canopy: technically, this means assigning, mapping, and transforming the network's output to desired MIDI events and sound generators

As we actively support the open-source software movement, which we find especially important for artists these days, we make the source code openly available¹.

2 PERFORMANCE

In our musical live performance, we collaboratively create experimental electronic music with a constantly evolving

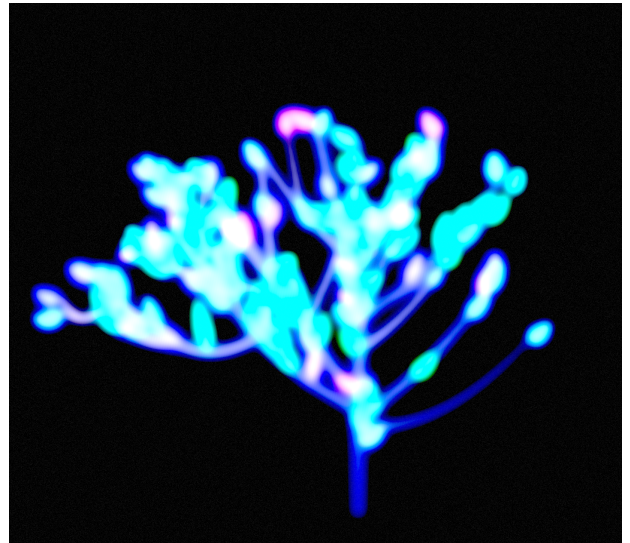


Figure 1: Screenshot of the live visualization of network topology and activations.

neural network. One programming human performer *A* initializes the neural network with random weights. The output of the neural network is mapped to MIDI events, creating a neural performer that starts sending random events. Those MIDI events are then sent to another human performer *B*, who manages the conversion of those events to actual audio. Using the digital synthesizer Vital, *B* starts to sculpt the resulting event stream by mapping MIDI *NoteOn* and *NoteOff* events as well as *control change* (CC) events to different synthesizers and internal control parameters, respectively. In turn, musician *A* reacts to the evolving soundscape, which creates a dynamic feedback loop.

The task of performer *A* is to seed the network. As we want the network to evolve during the performance, we

¹Code: <https://doi.org/10.5281/zenodo.21959142>

drew inspiration from *neuroevolution* concepts (Floreano et al., 2008). The network is not dense and layered, as modern neural networks typically are, but instead has a tree-like topology that itself presents an interesting artifact for visualization. Therefore, the growth of the neural network is visualized on a screen to the audience. A screenshot is presented in Figure 1.

In a sense, the two human performers and the organism as a neural performer become an improvising trio. At the same time, sharing the synthesizers’ parameter decisions with the neural network, and thereby granting it direct sound control, creates an interesting kind of creative collaboration that holds a lot of uncertainty and playfulness. Performer *A* tames the network at its root, while performer *B* tames the network’s random output.

To play in this constellation means to enter an open field of improvisation, to build on one’s own musical knowledge and experience to co-create patterns with the machine. It means listening, reacting, and shaping. Our performance does not take this creative practice away from the musician. Rather, it opens up new possibilities within it: new patterns, new musical structures, new harmonic relationships, new rhythmic figures, new timbres, new melodies. Crucially, our neural network does not generate music at the click of a button. It *needs* human co-musicians.

Accessibility and openness are also important concerns. Our performance is designed to work across a wide range of contexts, from complex environments such as Super-Collider or Pure Data, to software synthesizers like Vital or Surge XT, to samplers like Decent Sampler and FluidSynth, to any analog synthesizer with a MIDI-compatible input. By prioritizing free and open-source tools, we want to demonstrate that meaningful human-machine music-making does not require proprietary or resource-intensive infrastructure and can be accomplished with any MIDI-based musical gear, digital and analog.

3 BACKGROUND

Inspired by Tarkovsky’s *Sculpting in Time* (Tarkovsky, 1986), where the director argues that film does not merely show events, but captures lived time, memory, and duration, we treat the live feedback loop not as data processing, but as the shaping of a living present. Our neural network does not replace the artist, it acts as a catalyst that allows us to carve musical structures directly from the flow of time. For independent musicians, it becomes increasingly important not to feed their ideas and creativity into large, company-owned datasets. This is why we prefer AI models that are lightweight and energy-efficient, and evolve in real time, together with the human musicians, without permanently storing or exporting any data.

Following traditions from the field of *human-computer interaction* (HCI), a growing body of work has shifted focus away from fully autonomous generation and toward systems that foreground human agency. Ben-Tal and Dolan’s *Musical and Meta-Musical Conversations* (Ben-Tal and Dolan, 2023) documents a duo-improvisation practice between pianist and artificial improviser that deliberately avoids large-scale machine learning in favor of real-time machine listening, using MIR techniques to extract pitch, rhythm, and timbre from the live acoustic signal and respond in the moment. As the authors reflect, the system “does not employ the latest machine learning techniques with their reliance on extensive data and high-end com-

putational power,” deploying instead lighter, rule-based and statistical approaches in musically inventive ways. We follow this direction, and let go of optimization objectives altogether, trying to manually map ever-evolving neural behavior in a live setting.

A complementary conceptual framework is offered by Ted Moore’s *Musical Agents, Agency, & AI: Towards a Phenomenological Understanding* (Moore, 2024), which proposes that a technological system is perceived as a genuine collaborative agent when it is experienced as separate from the user, capable of surprise, intended as a collaborator, and able to mirror the user’s own musical intentions. This phenomenological approach suggests that what constitutes AI in music should be understood through perception rather than through technological implementation alone. This is a framing that resonates with our no-objective design philosophy.

Our artist-controlled AI tooling also connects to Rebecca Fiebrink’s foundational work on the Wekinator (Fiebrink, 2021), an open-source software for real-time interactive machine learning that has become a key reference point in the field, demonstrating that lightweight, human-controlled tools can sustain rich creative practice. Similarly, Wang and Martin’s *Off-the-shelf* explores live improvisation between a human electronic musician and a generative AI system controlling a compact synthesizer, foregrounding “minimalism in both technology and aesthetics” as a design principle in human-machine music-making (Wang and Martin, 2024). Also, Saint-Germier, Canonne, and Fiorini examined questions of embodiment in the co-improvisation software Somax2, investigating how machine-learning-based systems can engage meaningfully with human performers beyond mere technical responsiveness (Saint-Germier et al., 2024).

4 IMPLEMENTATION

The implementation of our work is based on the BbMuse framework (Ostermann, 2026). BbMuse is a blackboard-driven framework (Hayes-Roth, 1985) for real-time interactive music systems. It allows for fast prototyping of ideas and is well suited both for a proper implementation of some variant of *neuroevolution* (Floreano et al., 2008; Stanley et al., 2019) and for real-time interaction and even live coding (Collins et al., 2003). The neuroevolution strategy is loosely inspired by NEAT (Stanley and Miikkulainen, 2002). We grow branches of neural connections, while the set of neurons with no further forward connection (leaves) forms the output of the system.

The BbMuse project, which by the nature of BbMuse is highly modular and data-flow-based, is depicted in Figure 2. Modules declare which representations they *require*, *use*, and *provide*; the graph is derived from these declarations. Our system consists of five modules (*HumanInput*, *Controller*, *Prober*, *MIDISender*, *Visualizer*) operating on three internal representations (*Commands*, *NeuralTree*, *MIDIMessages*) and two external sinks (an *ext. MIDI device* and the *Godot Engine*).

Human performer *A* acts through the *HumanInput* module, which writes into the *Commands* representation. The released code ships a stand-in that samples the commands from Bernoulli distributions roughly once per second, so that the system can be run without a performer. In the performance itself, this module is live coded or a simple GUI is used. There are exactly four commands, and they

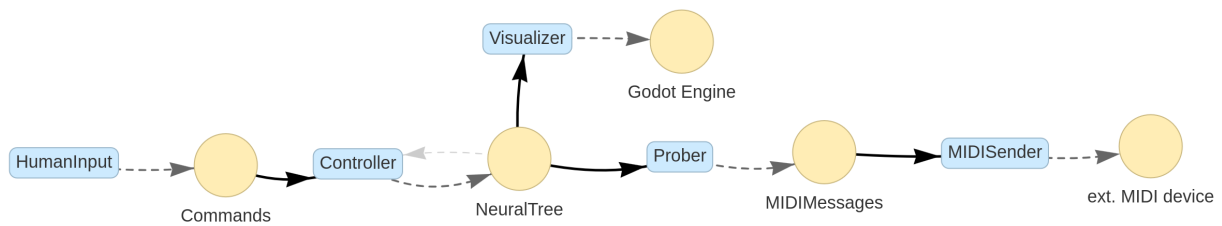


Figure 2: The BbMuse system architecture and its connection to external devices. Modules are blue, data representations are yellow. Created with BbMuse’s automatic visualization feature to inspect the dependency graph.

are the only way the network ever changes: *mutate_neuron* perturbs one weight of a randomly chosen neuron by a normally distributed offset and with small probability also its bias. *thicken* appends a neuron to a randomly chosen inner node, widening it and extending the input weights of all its children. *new_leaf* attaches a new leaf to a randomly chosen inner node. and *grow_branch* takes a random leaf, inserts a new inner node between it and its old parent, hands the leaf’s neurons over to that node, and gives the leaf a fresh set of neurons. The *Controller* consumes these commands, applies them to the *NeuralTree*, and re-evaluates the network afterwards.

The *NeuralTree* is not an MLP with layers, but a dynamic structure that grows from a single root. The only constraint we imposed, mostly to allow for the tree-like visualization, is that every node has exactly one parent (except the root), which makes the graph feed-forward and acyclic by construction. Inner nodes use ReLU activations, while leaves, i.e. the nodes with no further forward connection, use sigmoid activations and always carry exactly two output neurons. These two values in $[0, 1]$ form the output of the neural tree. The root is fed with a constant input, so the network is a deterministic function of its own topology and weights: the output only changes when the tree mutates. There is no fitness function and no learning signal. The variation is driven entirely by human decisions.

The *Prober* then collects the activations of all leaves and maps them to MIDI data. It fires at irregular intervals drawn from a skewed random distribution whose scale shrinks with the number of leaves, so a growing tree produces an increasingly dense event stream. Per firing, one random leaf is turned into a *NoteOn* event, with its first activation scaled to a MIDI note number (0–127) and its second to a MIDI velocity (0–127). *NoteOff* events are not tracked but fired blindly at random pitches, in a quantity calibrated such that the expected polyphony grows only sublinearly with the number of leaves. Without that, the tree would quickly saturate all 128 pitches. Every leaf additionally emits a CC event, its index determining the controller number, its two activations the mean μ and the standard deviation σ of a normal distribution from which the value is sampled, resulting in constantly changing CC values that slightly shift around the mean and only jump when neural weights in the path of its forward pass are mutated. Voices are spread over multiple channels. The resulting message list is shuffled so that a performer can still catch distinct events in a DAW that provides an interactive MIDI CC learning feature (without shuffling, single CC numbers consistently arrived first and monopolized the learn function). The *MIDISender* finally flushes the MIDI event queue to a MIDI port from where human performer *B* drives software and/or hardware synthesizers and samplers with the provided MIDI stream.

The whole point of this methodology is for human performer *A* (see Section 2) to try to influence the audio stream created by performer *B* in a way that *A* likes. However, the performers will realize that a chosen action in this deliberately chaotic system often does not lead to the expected behavior but to a wide range of unforeseen outcomes, which creates an experience we find enjoyable in itself. This steered behavior that cannot be fully controlled is part of the whole concept of our performance as discussed in Sections 1 and 2.

5 VISUALS

We applied our *tree* metaphor directly to visual feedback for the audience, and as inspiration for the human performers: the topology of the acyclic graph is rendered as a growing tree (see Figure 1) using the Godot Engine (Linietsky et al., 2014), in a hue-jittering blue palette on black, where the number of neurons of a node is mapped to the thickness of the corresponding branch. The *Visualizer* module serializes the graph as a flat list of nodes with parent, children, and thickness, and streams it as newline-delimited JSON over a TCP connection from a background thread. In a simple setup, we run both programs on the same local machine and send the tree data via the loopback address.

On the Godot side, the data passes through three stages: *Layout* rebuilds the tree from the parent pointers, hangs a synthetic ground node below the root so that the trunk becomes an ordinary edge, and assigns each branch an angle. The available opening angle is distributed among siblings by subtree size and thickness and pulled slightly towards the vertical. Two details keep the tree visually stable while it grows: all irregularities (angular jitter, leaf size, attachment point) are derived from a hash of the node id and are therefore constant across updates, and the oldest child of a node keeps the tip of the parent branch while later siblings attach further down. Without this, every new leaf would visibly rearrange its siblings. *Simulation* adds growth and motion. Positions are not integrated but constructed: only the angular deviation of a branch from its rest angle is simulated, and these deviations accumulate from parent to child, so the trunk barely moves while the tips move most and a gust visibly travels through the tree. Stiffness, damping, and wind sensitivity are interpolated by depth, wind is sampled from a spatially coherent noise field, and new layout targets are eased in rather than applied instantly, so that a structural change never causes a jump. New nodes, in contrast, adopt their target immediately and grow out. *Renderer* finally draws each edge as a quadratic Bézier with a width profile colored from trunk to tip by depth, and continuously rescales the whole drawing so that the tree stays inside the viewport as it grows, with the trunk anchored at the bottom center.

The image is then passed through two full-screen shader stages. The first is a glitch stage with per-channel offset, horizontal block displacement, and a short jump of the whole frame. It is driven by the data rather than by a timer, i.e., every new snapshot from the network triggers a pulse that decays exponentially, so that each act of performer A is also felt visually. The second stage emulates a screen being filmed with a hand-held phone camera, in the order of a real signal path: shake and breathing zoom, optical softness with chroma smear, moiré and rolling-shutter bands, saturation and contrast, lifted blacks, vignette, and film grain. Both stages are written to preserve values above 1.0, so that the bloom applied at the very end still reacts to the bright parts of the tree (using HDR rendering). The result is a picture that never sits still and that hides the discrete, graph-like nature of the underlying structure behind an apparently organic surface.

REFERENCES

- Ben-Tal, O. and Dolan, D. (2023). Musical and meta-musical conversations. In *Proceedings of the International Conference on AI and Musical Creativity (AIMC 2023)*.
- Collins, N., McLean, A., Rohrhuber, J., and Ward, A. (2003). Live coding in laptop performance. *Organised Sound*, 8(3):321–330.
- Fiebrink, R. (2021). The Wekinator and real-time interactive machine learning for creative practice. Keynote, International Conference on AI and Musical Creativity (AIMC 2021).
- Floreano, D., Dürr, P., and Mattiussi, C. (2008). Neuroevolution: From architectures to learning. *Evolutionary Intelligence*, 1(1):47–62.
- Hayes-Roth, B. (1985). A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321.
- Linietzky, J., Manzur, A., and Godot Engine contributors (2014). Godot Engine. <https://godotengine.org/>. Open-source game engine.
- Moore, T. (2024). Musical agents, agency, & AI: Towards a phenomenological understanding. In *Proceedings of the International Computer Music Conference (ICMC 2024)*, Seoul, South Korea.
- Ostermann, F. (2026). BbMuse: A blackboard-driven framework for real-time interactive music. In *Proceedings of the International Computer Music Conference (ICMC 2026)*.
- Saint-Germier, P., Canonne, C., and Fiorini, M. (2024). Embodiment and co-improvisation with Somax2. In *Proceedings of the International Conference on AI and Musical Creativity (AIMC 2024)*.
- Stanley, K. O., Clune, J., Lehman, J., and Miikkulainen, R. (2019). Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35.
- Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.
- Tarkovsky, A. (1986). *Sculpting in Time: Reflections on the Cinema*. University of Texas Press, Austin, TX.
- Wang, B. and Martin, C. P. (2024). Off-the-shelf: Live improvisation with a generative AI and a compact synthesizer. In *Proceedings of the International Conference on AI and Musical Creativity (AIMC 2024)*.