# Minimal Criterion Artist Collective

Kai Arulkumaran
ARAYA Inc.
Tokyo, Japan
kai_arulkumaran@araya.org

Thu Nguyen-Phuoc
University of Bath
Bath, United Kingdom
t.nguyen.phuoc@bath.ac.uk

## ABSTRACT

Minimal criterion co-evolution (MCC) is an evolutionary algorithm that uses a simple reproduction constraint between two interacting populations to drive an open-ended search process. While it has previously been applied to parameterise simple agents and environments, in this work we extend its use to the generation of art: synthesising both images and music. As a creative AI tool which does not require any data, the use of MCC emphasises the design of the creative medium.

## CCS CONCEPTS

• **Applied computing** → **Media arts**; • **Computing methodologies** → **Genetic algorithms**; *Neural networks.*

## KEYWORDS

open-ended evolution, co-evolution, generative art

**ACM Reference Format:**
Kai Arulkumaran and Thu Nguyen-Phuoc. 2022. Minimal Criterion Artist Collective. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3520304.3528763

## 1 INTRODUCTION

Open-endedness is a concept originating from the abandonment of the traditional objective in machine learning, and instead replacing it with a search for *novelty* [13]. Inspired by evolution, open-ended algorithms typically involve the evolution of a population, where novelty is defined in the space of their "behaviours". In the common case of autonomous agents, a typical hand-engineered "behavioural descriptor" used is the final location of the agent—encouraging exploration within an environment.

Similarly to traditional objective functions, hand-engineered behavioural descriptors limit the scope of novelty search [19]. Once again, the natural process of evolution provides a solution in the form of co-evolution, in which the objective is implicitly defined via the interaction between two populations [18]. In this work, we build upon a recent open-ended co-evolutionary algorithm, minimal criterion coevolution (MCC) [3]. In MCC, members of the two populations (which we will henceforth refer to as "generators" and "discriminators" [8]), are allowed to reproduce if they fulfil a "minimal criterion" [15] based on successfully interacting with a member of the opposite population. Broadly speaking, a generator must produce an output that is accepted by a discriminator, and vice versa. Combining MCC with resource limitation [4], in which discriminators can only accept a limited number of generators, futher promotes divergent solutions.

**Algorithm 1** MCC with resource limitation for art

```
Require: numSeeds ≥ 1                          ▷ Size of initial viable population
  maxGenerations ≥ 1                           ▷ Max number of generations
  numParents ≥ 1                               ▷ Number of parents per generation
  resourceLimit ≥ 1                            ▷ Max number of evaluations per solution
  randPop ← generateRandomPopulation()
  viablePop ← evolveSeedGenomes(randPop, numSeeds)
  for i = 0; i < maxGenerations; i++ do
      incrementAge(viablePop)
      parents ← viablePop.popLeft(numParents)
      children ← reproduce(parents)
      viablePop.append(parents)
      generators ← filter(children, isGenerator)
      discriminators ← filter(children, isDiscriminator)
      for child in children do
          if isGenerator(child) then
              for discriminator in discriminators do
                  if discriminator.usage < resourceLimit then
                      evalIndiv ← discriminator
                      break
                  end if
              end for
          else
              evalIndiv ← next(generators)
          end if
          mcSatisfied ← evaluateMC(child, evalIndiv)
          if mcSatisfied then
              incrementUsage(filter([child, evalIndiv], isDiscriminator))
              viablePop.append(child)
          end if
      end for
      if viablePop.size > viablePop.capacity then
          numRemovals ← viablePop.size − viablePop.capacity
          viablePop ← removeOldest(viablePop, numRemovals)
      end if
  end for
```

Ironically, despite the potential for open-ended algorithms to be applied to domains without a clear objective, they tend to still be applied to task-oriented domains. For instance, prior work on MCC focuses on populations of maze generators and maze solvers[1] [3, 4]. We propose that one of the largest potentials of open-ended algorithms lies in applications to art, with the goal of producing a continuously evolving set of novel artworks.

Inspired by Picbreeder [22], we propose using MCC, collaboratively with human curation, as a tool for generative art [1]. As opposed to defining a mathematical formula to characterise novel "art", we instead build inductive biases into the generator and discriminator populations, and let the populations interact via MCC. To demonstrate the generality of MCC for generative art, we showcase image generations from 4 different types of generators, as well as audio generations. In contrast to popular methods for generative art,

[1] Our discriminators and generators, respectively.


*GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA*

ACM ISBN 978-1-4503-9268-6/22/07.
https://doi.org/10.1145/3520304.3528763

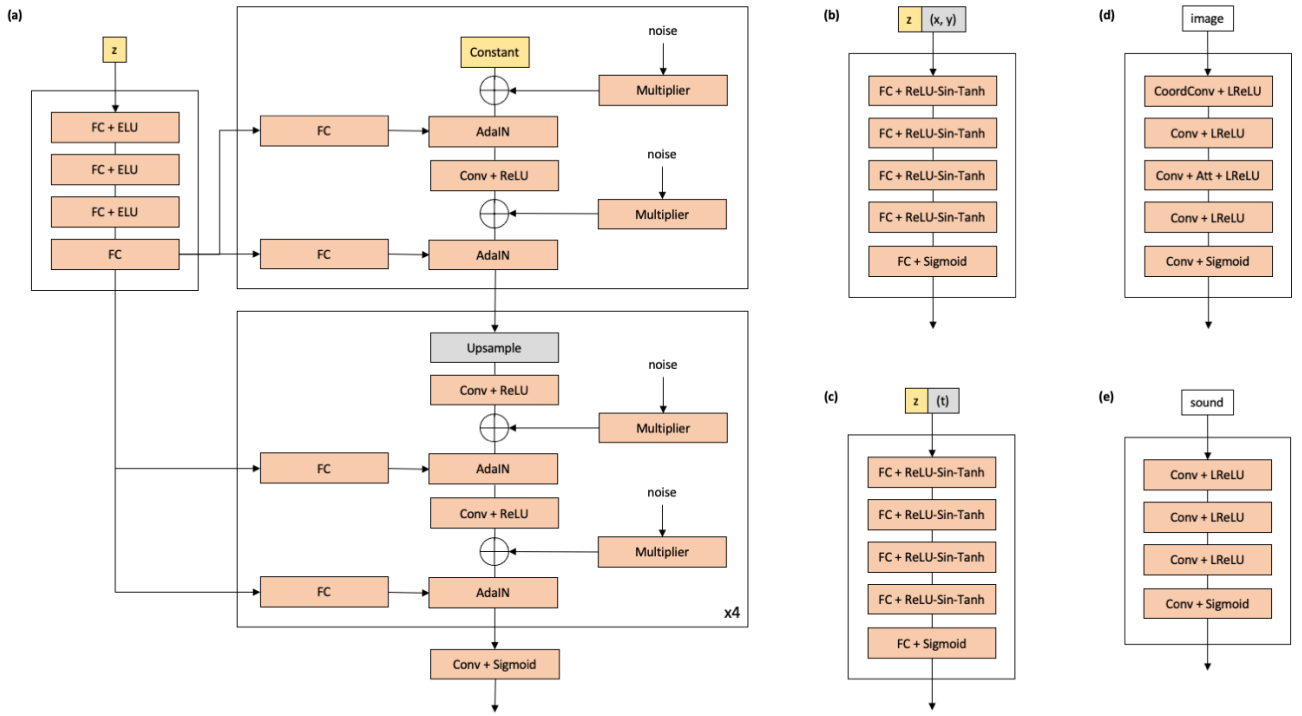


**Figure 1: Architectures: (a) StyleGAN generator for images; (b) CPPN generator for images; (c) CPPN generator for audio; (d) DCGAN + self-attention discriminator for images; (e) 1D convolution discriminator for audio. Parameters are shaded in yellow/orange; fixed operations/embeddings are shaded in grey. The StyleGAN blocks use bilinear upsampling, adaptive instance normalisation (AdaIN) and random Gaussian noise [11]. The CPPNs use a mix of ReLU, sine and tanh activation functions.**

such as DeepDream [16] and neural style transfer [7], our algorithm does not require any data/pretrained models, and can be applied to arbitrarily parameterised generative algorithms. Source code and samples are available at https://github.com/Kaixhin/MCAC.

## 2 METHODS

The high-level algorithm (Algorithm 1) is essentially the same as in prior work [4], and differs in the implementation details. This is a testament to the generality of the MCC algorithm.

### 2.1 Generators

There has been a concerted effort in recent years to develop machine learning models that can generate plausible images, across a range of domains. In particular, deep generative models [2] have achieved impressive results in both conditional and unconditional image generation. One of the most popular frameworks is that of generative adversarial networks (GANs) [8], in which a generator network takes a latent vector $z$ as input and produces an image as output; it is trained to "fool" a discriminator network which takes both fake and real images as input and produces a binary classification as output. We use network architectures developed in this line of work, as the generators produce images in one step, without requiring intermediate latent variables, sampling, or optimisation during inference. We experimented with a range of network architectures, starting with the DCGAN architecture [20], plus CoordConv [14] and self-attention layers [26], for both the generators and discriminators, but found that the generators primarily produced high frequency patterns. By switching to a StyleGAN generator [11] (Figure 1a), which progressively builds up an image through upsampling operations, we were able to generate images with larger spatial patterns.

For a different architectural bias, we used compositional pattern-producing networks (CPPNs) [24]. CPPNs take spatial coordinates as input and produce a single RGB value as output; producing an image requires evaluating the CPPN across a grid of coordinates. The activation functions used confer different spatial properties, such as discontinuities and periodicity. In contrast to vanilla CPPNs, we append a latent vector $z$ to the coordinate embeddings, allowing the generator to condition on this as well (Figure 1b).

Directly outputting pixel intensities allows for a largely unconstrained search space. One way to impose more control over the process is to have an intermediate synthesis step; we achieve this by modifying the CPPN-based generators to output a sequence of actions (e.g., brush size, pressure) for the `libmypaint` environment [6], allowing the generators to use a synthetic canvas and paintbrush. Its images are formed as composition of brushstrokes.

We are not restricted to neural-network-based generators either. Our fourth image generator is based on a simplified version of the Fractal Flame algorithm [5], in which points are plotted on a canvas based on the iterative application of a set of parameterised

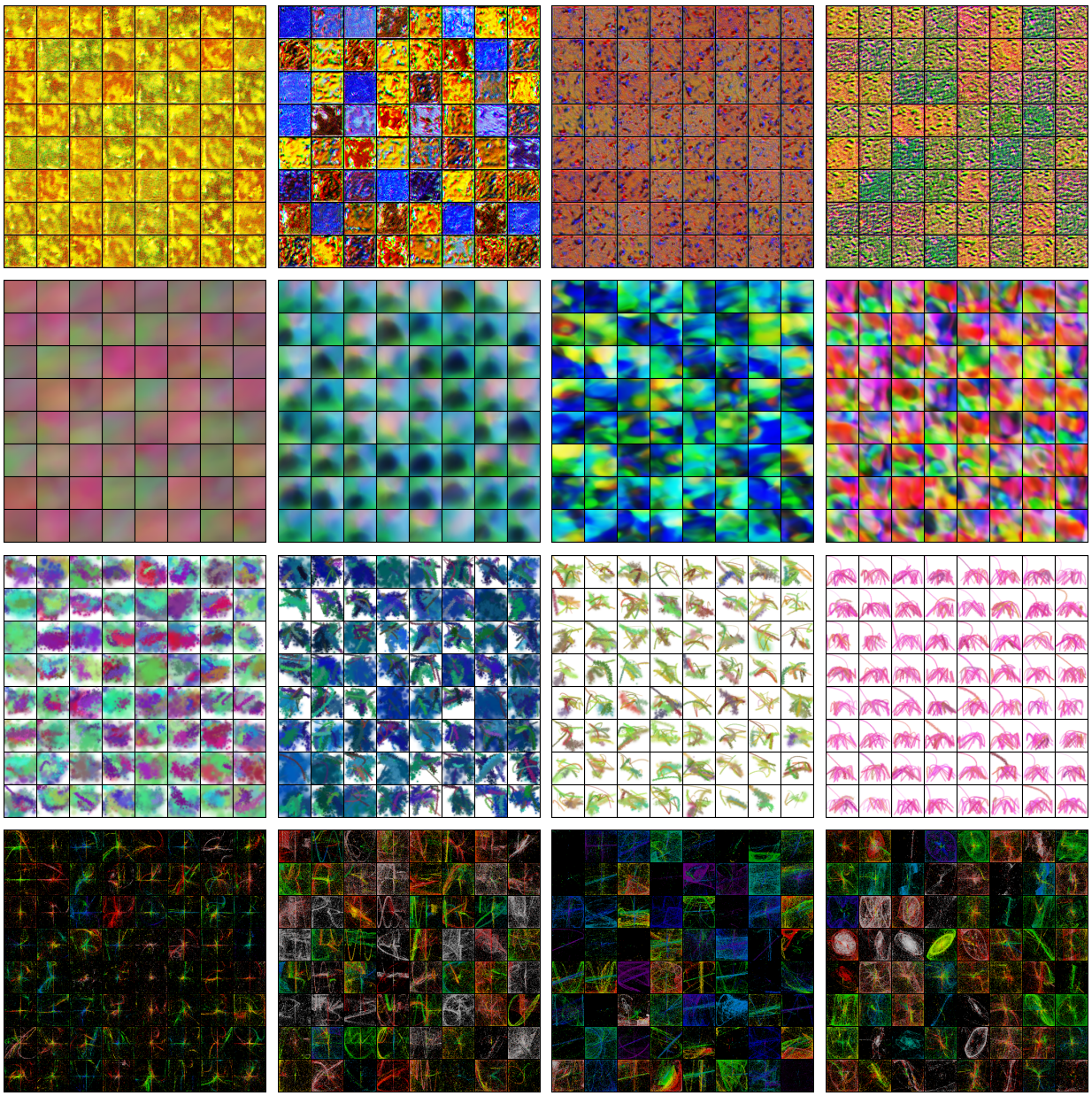

**Figure 2: Image generations: from top to bottom, StyleGAN, CPPN, `libmypaint` and Fractal Flame.**

nonlinear functions which take the current coordinate as input and output the next coordinate.

Finally, we modified the CPPN-based generators to produce note specifications for MIDI files, with the audio timestep replacing the spatial coordinates in the input (Figure 1c). Given a synthesiser/samples, this also allows the algorithm to generate audio.

## 2.2 Discriminators

We used the modified DCGAN architecture (Figure 1d) for evaluating images and a simple 1D convolutional network for evaluating audio (Figure 1e). Of note is that all discriminators output a scalar $\in (0, 1)$. Although this no longer has a semantic meaning of "real" vs. "fake" [8], we use this property for our MC.

## 2.3 Minimal Criterion

When applying the MC to generating art, it is unclear what the criterion itself should be. While one could potentially use a function based on the outputs of models trained on large datasets, such as GAN discriminators or (vision-based) object detectors, in this work we rely on the priors built into convolutional neural networks [25]: the outputs of the discriminators will vary based on properties of the input that have some correlation with human perception.

Our MC is to have the generator force the output of the discriminator to vary across a batch of inputs, i.e., to have the standard deviation of the output be over a handpicked threshold. In order to generate a batch of outputs, we evaluate generators by perturbing their latent parameter $z$ (for the Fractal Flame algorithm, $z$ is the set of all parameters [5]) by multiplying it by a noise vector of the size of $batch\_size \times |z|$. Like an artist collective with shared aesthetics, a single generator (with parameters shared across a batch) can thereby produce a collection of artworks whenever it is evaluated. The sharing of parameters makes this a nontrivial criterion [23], as otherwise diverse outputs could be produced trivially by using separate parameters per output (within a batch).

### 2.4 Reproduction

The parameters for all neural-network-based generators use the default weight initialisations in PyTorch [17]. For Fractal Flame, the function weights and parameters are sampled from a normal distribution, while the colours are sampled from a uniform distribution $\in [0, 1]$. During the reproduction step we mutate all parameters by multiplying by isotropically-scaled Gaussian noise; Fractal Flame parameters are then normalised/clamped as necessary. Although the MCC algorithm specifies creating a viable population before starting the full process, in practice we did not need to do so.

## 3 RESULTS

As seen in Figure 2, the design of the generator has a large impact on the types of images produced. Notably, the criterion that a discriminator's output has a minimum standard deviation does not necessitate that the range of outputs from a single generator will be subjectively diverse from a human perspective. The current implementation of our algorithm is most useful for finding interesting parameterisations of more constrained generative models, i.e., the Fractal Flame algorithm; MCC tends to filter out models that output sparse, highly similar, or very noisy images.

## 4 DISCUSSION

We frame a discussion of our work under the concepts of evolutionary art delineated by Romero & Machado [21]. The inductive biases within our generators determine the possible "solution spaces"—generality has more potential, but also reduces the average "fitness" of solutions. Constraining the generators to use artificial tools, such as paintbrushes or synthesisers, seems to yield a good balance between expressitivity and aesthetics. Likewise, the discriminators determine the "fitness landscape"—although with MCC there is no explicit fitness, our goal is to generate images or audio that are appealing to humans. Having a human-in-the-loop at every step, like in Picbreeder [22], requires significant interaction. While heuristic perceptual criteria have been developed in the field of AI [9, 12], we expect psychophysics-derived models trained on human preferences [10] to be the most promising direction for improving the search space and manual curation process. Importantly, art is subjective, which would necessitate taking into account the multimodal nature of preference profiles [10]. Quantitative metrics can also be used to measure diversity, though we note that the generative process may not necessarily work best for the curator by explicitly maximising either intra- or inter-generator diversity. That said, with some manual control we believe the use of MCC to be promising for "controlling diversity"—with some intuition about appropriate mediums and a small amount of compute, we were able to synthesise a variety of interesting outputs with limited tuning. While the era of a completely autonomous artist might be further off, experienced generative artists could well benefit from the use of open-ended algorithms to explore novel artworks.

## REFERENCES


[1] Margaret A Boden and Ernest A Edmonds. 2009. What is Generative Art? *Digit. Creat.* 20, 1-2 (2009), 21–46.
[2] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G Willcocks. 2021. Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models. *arXiv:2103.04922* (2021).
[3] Jonathan C Brant and Kenneth O Stanley. 2017. Minimal Criterion Coevolution: A New Approach to Open-ended Search. In *GECCO*.
[4] Jonathan C Brant and Kenneth O Stanley. 2020. Diversity Preservation in Minimal Criterion Coevolution Through Resource Limitation. In *GECCO*.
[5] Scott Draves and Erik Reckase. 2008. *The Fractal Flame Algorithm*. Technical Report. Spotworks.
[6] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Ali Eslami, and Oriol Vinyals. 2018. Synthesizing Programs for Images Using Reinforced Adversarial Learning. In *ICML*.
[7] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. In *CVPR*.
[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NeurIPS*.
[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-scale Update Rule Converge to a Local Nash Equilibrium. In *NeurIPS*.
[10] Kiyohito Iigaya, Sanghyun Yi, Iman A Wahle, Koranis Tanwisuth, and John P O'Doherty. 2021. Aesthetic Preference for Art Can Be Predicted From a Mixture of Low- and High-level Visual Features. *Nat. Hum. Behav.* 5, 6 (2021), 743–755.
[11] Tero Karras, Samuli Laine, and Timo Aila. 2019. A Style-based Generator Architecture for Generative Adversarial Networks. In *CVPR*.
[12] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. 2018. Fréchet Audio Distance: A Metric for Evaluating Music Enhancement Algorithms. *arXiv:1812.08466* (2018).
[13] Joel Lehman and Kenneth O Stanley. 2008. Exploiting Open-endedness to Solve Problems Through the Search for Novelty. In *ALIFE*.
[14] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. 2018. An Intriguing Failing of Convolutional Neural nNtworks and the CoordConv Solution. In *NeurIPS*.
[15] Claudio Mattiussi and Dario Floreano. 2003. *Viability Evolution: Elimination and Extinction in Evolutionary Computation*. Technical Report. Ecole Polytechnique Fédérale de Lausanne.
[16] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. 2015. Inceptionism: Going Deeper into Neural Networks. https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html
[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, et al. 2019. Pytorch: An Imperative Style, High-performance Deep Learning Library. In *NeurIPS*.
[18] Elena Popovici, Anthony Bucci, R Paul Wiegand, and Edwin D De Jong. 2012. *Coevolutionary Principles*. Springer, 987–1033.
[19] Justin K Pugh, Lisa B Soros, Paul A Szerlip, and Kenneth O Stanley. 2015. Confronting the Challenge of Quality Diversity. In *GECCO*.
[20] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *ICLR*.
[21] Juan J Romero and Penousal Machado. 2008. *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*. Springer Science & Business Media.
[22] Jimmy Secretan, Nicholas Beato, David B D Ambrosio, Adelein Rodriguez, Adam Campbell, and Kenneth O Stanley. 2008. Picbreeder: Evolving Pictures Collaboratively Online. In *SIGCHI*.
[23] L Soros and Kenneth O Stanley. 2014. Identifying Necessary Conditions for Open-ended Evolution Through the Artificial Life World of Chromaria. In *ALIFE*.
[24] Kenneth O Stanley. 2007. Compositional Pattern Producing Networks: A Novel Abstraction of Development. *Genet. Program. Evolvable Mach.* 8, 2 (2007), 131–162.
[25] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2018. Deep Image Prior. In *CVPR*.
[26] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2019. Self-attention Generative Adversarial Networks. In *ICML*.