Adversarial Image Generation using Evolution and Deep Learning

Jacob Soderlund and Alan Blair
School of Computer Science and Engineering
University of New South Wales
Sydney, 2052 Australia
Email: blair@cse.unsw.edu.au

***Abstract*—There has recently been renewed interest in the paradigm of artist-critic coevolution, or adversarial training, in which an artist tries to generate images which are similar in style to a set of real images, and a critic tries to discriminate between the real images and those generated by the artist. We explore a novel configuration of this paradigm, where the artist is trained by hierarchical evolution using an evolutionary automatic programming language called HERCL, and the critic is a convolutional neural network. The system implicitly solves the constrained optimization problem of generating images which have low algorithmic complexity, but are sufficiently suggestive of real-world images as to fool a trained critic with an architecture loosely modeled on the human visual system. The resulting images are not necessarily photorealistic, but often consist of geometric shapes and patterns which remind us of everyday objects, landscapes or designs in a manner reminiscent of abstract art. We explore the coevolutionary dynamics between artist and critic, and discuss possible combinations of this framework with interactive evolution or other human-in-the-loop paradigms.**

I. Introduction

There has recently been renewed interest in the paradigm of artist-critic coevolution or adversarial training in which an artist tries to generate images which are similar in style to a set of "real" images, and a critic tries to discriminate between the real images and those generated by the artist (Fig. 1).

The earliest work in this area followed an interactive evolution scenario, with a human playing the role of the critic, and the artist trained by some form of evolutionary computation such as biomorphs [3], Genetic Programming [21] and, latterly, Cellular Automata [9] or Convolutional Pattern Producing Networks [18]. In these systems, several candidate images appear on the screen and the user is invited to select one or more of them for inclusion in the next generation. These approaches have produced some remarkable images, but the process can be time-consuming for the human as several dozen generations are often required in order to produce a pleasing image.

In recent years, attempts have been made to replace the human with a fully automated critic such as a Self Organizing Map [20] or a 2-layer neural network trained on certain statistical features extracted from the image [14, 7, 13]. The critic is rewarded for its ability to distinguish between real images and those generated by the artist, while the artist is rewarded for producing images that fool the critic.

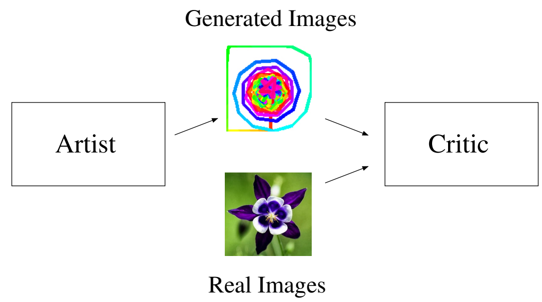

Fig. 1: Artist-Critic Coevolution

TABLE I: Paradigms for Artist-Critic Coevolution

Artist	Critic	Method	Reference
Biomorph	Human	Blind Watchmaker	(Dawkins, 1986)
GP	Human	Interactive Evolution	(Sims, 1991)
CPPN	Human	PicBreeder	(Secretan, 2011)
CA	Human	EvoEco	(Kowaliw, 2012)
GP	SOM	Artificial Creativity	(Saunders, 2001)
GP	NN	Computational Aesthetics	(Machado, 2008)
Agents	NN	Evolutionary Art	(Greenfield, 2009)
GP	NN	Aesthetic Learning	(Li & Hu, 2010)
DCNN	DCNN	Generative Adversarial Nets	(Goodfellow,2014)
CPPN	DCNN	Convolutional Pattern Producing Nets	(Nguyen, 2015)
HERCL	HERCL	Co-Evolving Line Drawings	(Vickers, 2017)
HERCL	DCNN	HERCL Function / DCNN	(current work)

II. Methodology

In previous work [23] artists were evolved to produce line drawings using an evolutionary automatic programming language called HERCL; the critic was also a HERCL program, trained by hierarchical evolution [2]. Its inputs were statistical features from the image, similar to those used in [14]. Although the resulting images did have a certain naive charm about them, it seemed that the limited statistical features were not providing the critic with sufficient information to make an accurate determination.

Inspired by the remarkably realistic images that have recently been produced by Generative Adversarial Networks [6, 16] we decided to try a new variant where a Deep Convolutional Network plays the role of the critic. We adopt an approach similar to that of [15] where the artist (in our case, a HERCL program) acts as a function, taking as input

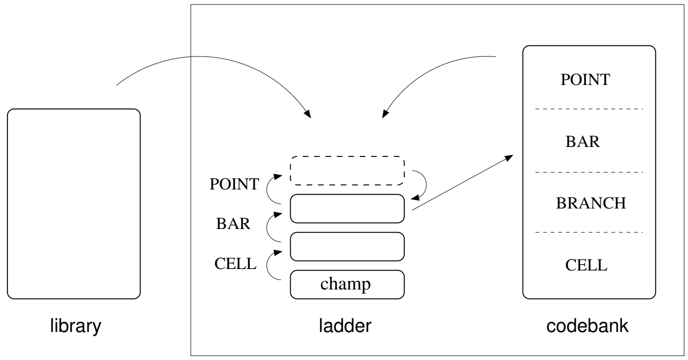

Fig. 2: Hierarchical evolutionary re-combination. If the top agent on the ladder becomes fitter than the one below it, the top agent will move down to replace the lower agent (which is transferred to the codebank). If the top agent exceeds its maximum number of allowable offspring without ever becoming fitter than the one below it, the top agent is removed from the ladder (and transferred to the codebank).

TABLE II: Modified LeNet-5 and All Convolutional Net

Modified LeNet-5 model
5×5 conv. × 6, Leaky ReLU
2×2 max pooling, stride 2
5×5 conv. ×16, Leaky ReLU, no padding
2×2 max pooling, stride 2
120 fully connected units, Leaky ReLU
84 fully connected units, Leaky ReLU
2 fully connected output units, Softmax

Modified All-Convolutional Net model
3×3 conv. × 96, Leaky ReLU, no padding
3×3 conv. × 96, Leaky ReLU, stride 2
3×3 conv. ×192, Leaky ReLU
3×3 conv. ×192, Leaky ReLU, stride 2
3×3 conv. ×192, Leaky ReLU
1×1 conv. ×192, Leaky ReLU
1×1 conv. × 10, Leaky ReLU
Global average pooling of each feature map
2 fully connected output units, Softmax

an x and y coordinate, and producing either a single value (for black and white images) or a triplet of values (for color images).

In each round, the critic (CNN) is trained by backpropagation to assign a value close to 0 for all "real" images, and close to 1 for the synthetic images produced in all previous rounds. Then, a HERCL program is evolved to produce an image for which the current critic will assign a value close to zero. This new image is added to the set of synthetic images, ready for the next round to begin. In other words, the system produces one image per round. In the first round, a blank image is used as the (single) synthetic image.

We tried two different CNN architectures, based on the LeNet-5 [11] and All Convolutional Net models [22] (see

TABLE III: HERCL Commands

<table>
<tr><td></td><td>Input and Output</td><td></td></tr>
<tr><td>i</td><td>fetch INPUT to input buffer</td><td></td></tr>
<tr><td>s</td><td>SCAN item from input buffer to stack</td><td></td></tr>
<tr><td>w</td><td>WRITE item from stack to output buffer</td><td></td></tr>
<tr><td>o</td><td>flush OUTPUT buffer</td><td></td></tr>
<tr><td></td><td>Stack Manipulation and Arithmetic</td><td></td></tr>
<tr><td>#</td><td>PUSH new item to stack</td><td>$\ldots\ldots \mapsto \ldots\ldots x$</td></tr>
<tr><td>!</td><td>POP top item from stack</td><td>$\ldots\ldots x \mapsto \ldots\ldots$</td></tr>
<tr><td>c</td><td>COPY top item on stack</td><td>$\ldots\ldots x \mapsto \ldots\ldots x, x$</td></tr>
<tr><td>x</td><td>SWAP top two items</td><td>$\ldots y, x \mapsto \ldots x, y$</td></tr>
<tr><td>y</td><td>ROTATE top three items</td><td>$z, y, x \mapsto x, z, y$</td></tr>
<tr><td>-</td><td>NEGATE top item</td><td>$\ldots\ldots x \mapsto \ldots\ldots (-x)$</td></tr>
<tr><td>+</td><td>ADD top two items</td><td>$\ldots y, x \mapsto \ldots (y+x)$</td></tr>
<tr><td>*</td><td>MULTIPLY top two items</td><td>$\ldots y, x \mapsto \ldots (y*x)$</td></tr>
<tr><td></td><td>Mathematical Functions</td><td></td></tr>
<tr><td>r</td><td>RECIPROCAL</td><td>$..\, x \to ..\, 1/x$</td></tr>
<tr><td>q</td><td>SQUARE ROOT</td><td>$..\, x \to ..\sqrt{x}$</td></tr>
<tr><td>e</td><td>EXPONENTIAL</td><td>$..\, x \mapsto ..\, e^x$</td></tr>
<tr><td>n</td><td>(natural) LOGARITHM</td><td>$..\, x \mapsto ..\, \log_e(x)$</td></tr>
<tr><td>a</td><td>ARCSINE</td><td>$..\, x \mapsto ..\, \sin^{-1}(x)$</td></tr>
<tr><td>h</td><td>TANH</td><td>$..\, x \mapsto ..\, \tanh(x)$</td></tr>
<tr><td>z</td><td>ROUND to nearest integer</td><td></td></tr>
<tr><td>?</td><td>push RANDOM value to stack</td><td></td></tr>
<tr><td></td><td>Double-Item Functions</td><td></td></tr>
<tr><td>%</td><td>DIVIDE/MODULO</td><td>$..\, y, x \mapsto ..\, (y/x), (y \bmod x)$</td></tr>
<tr><td>t</td><td>TRIG functions</td><td>$..\, \theta, r \mapsto ..\, r\sin\theta, r\cos\theta$</td></tr>
<tr><td>p</td><td>POLAR coords</td><td>$..\, y, x \mapsto ..\, \text{atan2}(y, x), \sqrt{x^2+y^2}$</td></tr>
<tr><td></td><td>Registers and Memory</td><td></td></tr>
<tr><td><</td><td>GET value from register</td><td></td></tr>
<tr><td>></td><td>PUT value into register</td><td></td></tr>
<tr><td>^</td><td>INCREMENT register</td><td></td></tr>
<tr><td>v</td><td>DECREMENT register</td><td></td></tr>
<tr><td>{</td><td>LOAD from memory location</td><td></td></tr>
<tr><td>}</td><td>STORE to memory location</td><td></td></tr>
<tr><td></td><td>Jump, Testing, Branching and Logic</td><td></td></tr>
<tr><td>j</td><td>JUMP to specified cell (subroutine)</td><td></td></tr>
<tr><td>|</td><td>BAR line (RETURN on .| HALT on 8|)</td><td></td></tr>
<tr><td>=</td><td>check register is EQUAL to top of stack</td><td></td></tr>
<tr><td>g</td><td>check register is GREATER than top of stack</td><td></td></tr>
<tr><td>:</td><td>if TRUE, branch FORWARD</td><td></td></tr>
<tr><td>;</td><td>if TRUE, branch BACK</td><td></td></tr>
<tr><td>&</td><td>logical AND</td><td></td></tr>
<tr><td>/</td><td>logical OR</td><td></td></tr>
<tr><td>~</td><td>logical NOT</td><td></td></tr>
</table>

Table II). The Adam optimizer was used [8] with softmax output and mean cross-entropy as the cost function.

The artist is evolved by Hierarchical Evolutionary Re-Combination [2] which maintains a small number of competing agents, arranged in a ladder, and preserves diversity through the use of a codebank (Fig. 2). Culled agents are transferred to the codebank where they remain available, for a period of time, as potential breeding partners. When a new agent is created, genetic material can be taken from the codebank, or from an external library. In the present work, the library contains the code from the final (successful) image of all preceding rounds, thus allowing the artist to create

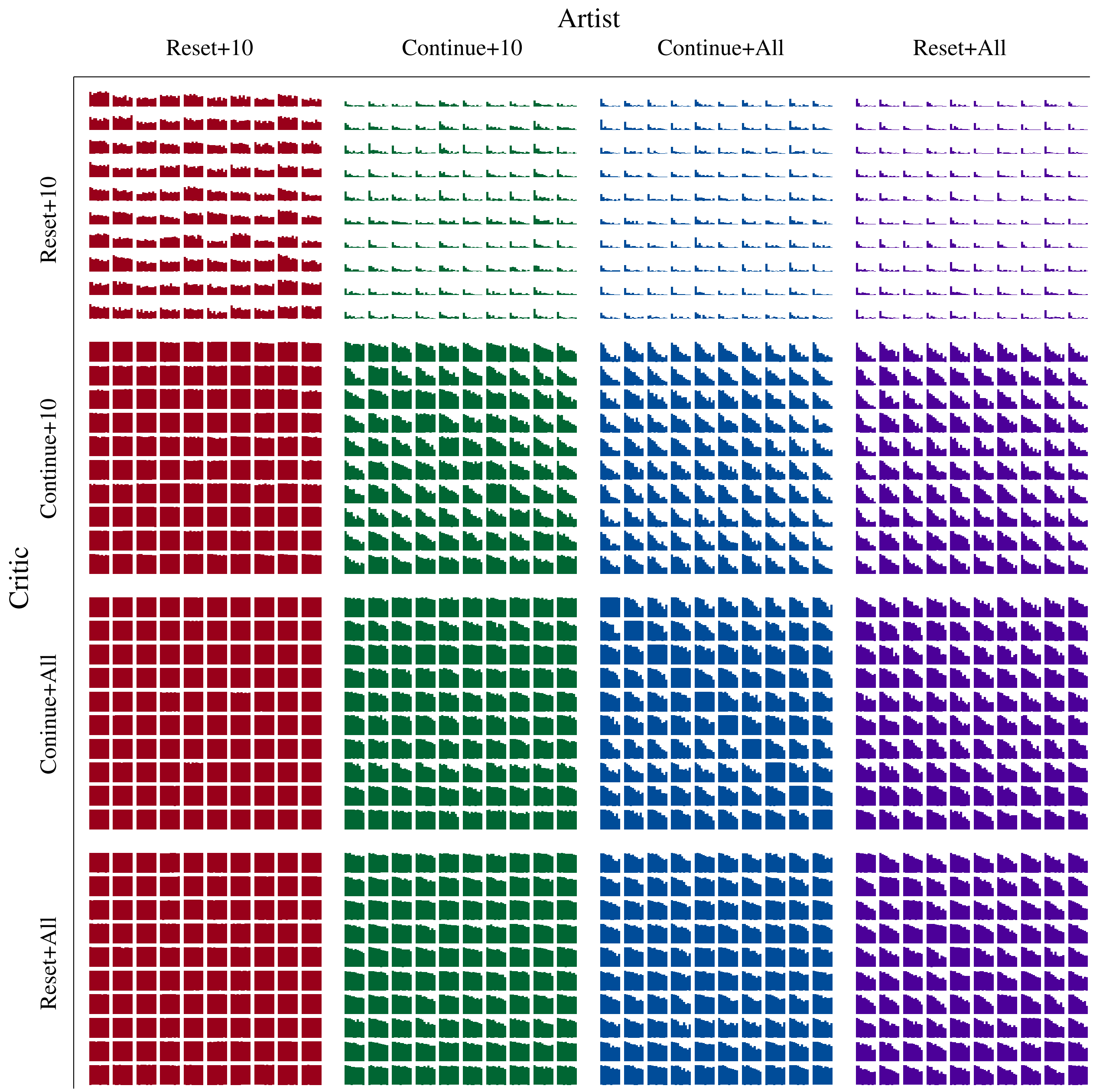

Fig. 3: Average cost for groups of 50 consecutive images in each run of the MNIST experiments (columns) assigned by final critic in each of the 40 runs (rows). Each tiny graph plots the average cost for 10 groups of 50 consecutive images in a run of 500 images. High cost is good for the critic; low cost is good for the artist.

new images by experimenting with variations on previously successful images.

In this adversarial context, the avoidance of over- or under-fitting can be thought of as maintaining a balance of power between artist and critic. If the critic is trained too long, it may assign values very close to $0\,(1)$ for the real (synthetic) images, respectively; if it is not trained long enough, it may assign values close to 0.5 for both the real and synthetic images. In either case, the artist may fail to achieve its target threshold. With this in mind, we trained each critic on the full set of $50\,000$ real images and an equal number of synthetic images (with repetition of image presentation taken into account). The HERCL programs were evolved until the cost function (value assigned by the critic) becomes lower than a threshold value of 0.01. In general, we found this provides a propitious balance between artist and critic. However, for some of the later runs, if the evolution exceeded a certain maximum number of evaluations without achieving its target threshold, the evolution was halted and the currently best-scoring image was accepted into the next round.

The entire coevolutionary system was tested using images from MNIST (Section III) and CIFAR-10 (Section IV).

III. MNIST and Coevolutionary Dynamics

Our first set of experiments used the MNIST dataset of handwritten digits. The artists were HERCL programs and were required to produce a single output for each pixel, to generate a gray-scale image. The critics used a modified LeNet-5 CNN architecture (see Table II).

Since adversarial training is a form of coevolution between artist and critic, the question arises as to whether there is some risk of the system getting stuck in a mediocre stable state, or a cycle in which the critic forgets how to classify earlier images, or the artist forgets how to fool earlier critics [17]. The fact that each critic is trained on the images produced by all previous artists should in theory help to prevent the system from falling into such a mediocre state or cyclic pattern. In earlier work with a simpler and faster critic [23] the fitness of each artist was a weighted average of the evaluation from all previous critics (with recent critics weighted more heavily). However, in the present scenario, this approach might be computationally prohibitive, because it would require several sets of CNN weights to be stored in GPU memory simultaneously. Therefore, we evaluate the fitness of each artist using only the current critic.

In order to explore the coevolutionary dynamics in more detail, we compare four different training variants:

Reset+10: The CNN weights are re-set to small random values at the beginning of each round. The CNN is trained only on images from the ten most recent rounds.

Continue+10: The final CNN weights from the previous round are used as initial weight values for training in the subsequent round. The CNN is trained only on images from the ten most recent rounds.

Continue+All: The final CNN weights from the previous round are used as the initial weight values for training in the subsequent round. The CNN is trained on images from all previous rounds.

Reset+All: The CNN weights are re-set to small random values at the beginning of each round. The CNN is trained on images produced in all previous rounds.

Ten runs were performed in each of the four variants, for 500 rounds, producing a total of 40 final critics and $40 \times 500 = 20\,000$ images.

A. Comparison of Artists

We can evaluate each image by computing the cost assigned to it by the final critic from each of the 40 runs. In most cases, the final critic from its own run will likely assign it a very high cost (since this image was included in its training set). But, for the other 39 critics, this will appear as a genuinely "unseen" image.

Each tiny graph in Fig. 3 plots the average cost for groups of 50 consecutive images in one run, assigned by the final critic in another run. These values are aggregated in Fig. 4, which plots the mean cost (expressed as a percentage) assigned by the final critics from all 40 runs, averaged over the 10 runs in each variant (and groups of 25 consecutive images from each run). To test the statistical significance of these results, Table IV shows the mean and SEM, over the final 200 images from the 10 runs in each variant, of the cost assigned by the final critics from the runs of another variant. (For Table IV, the cost assigned to an image by the final critic from its own run was omitted.)

We see that Continue+All and Reset+All (whose critics were trained on all preceding images) achieve significantly better cost than Reset+10 and Continue+10 (whose critics were trained only on images from the previous 10 rounds). Interestingly, Continue+10 performs considerably better than Reset+10. The Continue+10 critics can be thought of as undergoing one continuous training process, but with each image being dropped from the training set after 10 subsequent rounds. These preserved weights apparently provide a significant "residual" effect, with the network retaining its ability to reject earlier images (and others of the same style) for a considerable period of time after they have been removed from the training set.

Note also that Reset+All (where the weights were reset in each round) performed somewhat better than Continue+All (where the weights were preserved from one round to the next). The likely reason is that, although successive critics are

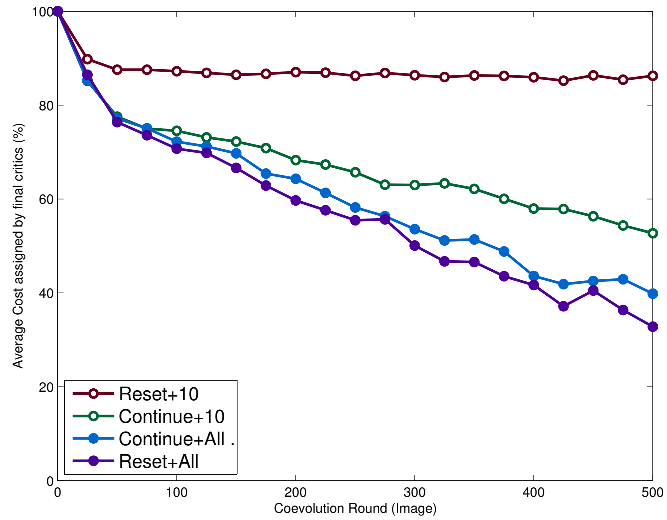

Fig. 4: Mean cost (%) assigned by the 40 final critics, averaged over 10 runs and 25 consecutive images in each run.

TABLE IV: Mean cost (%) for last 200 images in each run

Critic	Artist: Reset+10	Continue+10	Continue+All	Reset+All
Reset+10	45.3 ± 0.6	5.3 ± 0.4	3.4 ± 0.3	2.0 ± 0.3
Continue+10	97.8 ± 0.3	56.1 ± 0.6	35.1 ± 0.6	35.3 ± 0.6
Continue+All	99.9 ± 0.2	83.3 ± 0.5	63.9 ± 0.6	60.0 ± 0.6
Reset+All	99.3 ± 0.2	84.6 ± 0.4	75.1 ± 0.5	62.4 ± 0.5

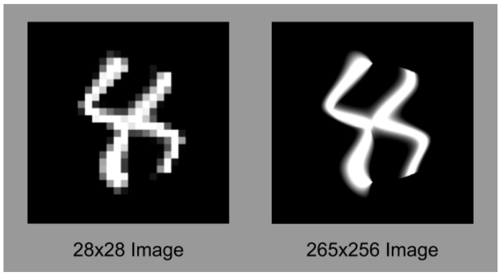

Fig. 5: HERCL image rendered at 28×28 resolution and at 256×256 resolution.

trained on very similar training sets (with only one additional image), the resetting of the weights provides extra diversity to the critics, in the same way that bagging or boosting methods are used to provide additional diversity in machine learning. Another possible explanation is that preserving the weights from one round to the next causes the network to give undue emphasis to the earlier (low quality) images, on which it has been trained the longest.

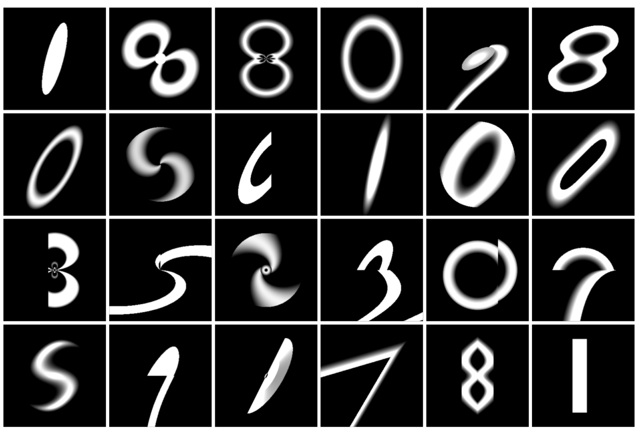

Fig. 6: Selected images generated on the MNIST dataset from all runs. Images were selected to show the diversity of recognizable images.

B. Subjective Analysis of Images

Using the artist as a function allows us to render the images at any desired resolution. For example, in Fig. 5, the 28×28 image on the left is the one that is fed to the critic; but, we are free to view the 256×256 image on the right if we find it more appealing.

We found that many of the images produced by our system do indeed look like digits, and that the images whose average cost from the final critics was lowest tended to be the most realistic (see Fig. 6). This is somewhat in contrast to [15] where it was reported that a CPPN artist tended to produce very un-digit-like images which nevertheless fooled the critic. The difference might be due to the fact that our hierarchical evolution paradigm includes selective pressure towards shorter programs. Images from the early rounds tend to be very regular

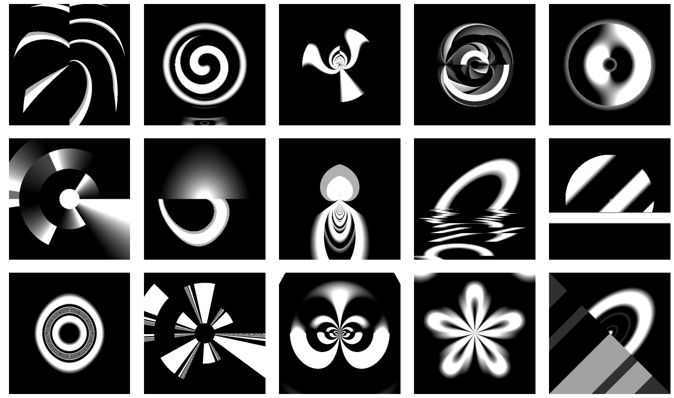

Fig. 7: Images from the MNIST runs which do not resemble digits, but might nevertheless have some artistic merit.

geometric shapes (circles and lines) which resemble simple digits like 0 and 1. In later images, the shapes tend to be slightly irregular and therefore more natural looking. This is presumably because the critic has figured out how to reject the regular images so the artist must add extra complexity to its program in order to fool the new critic.

There were also images which do not resemble digits but may have some intrinsic artistic merit (see Fig. 7).

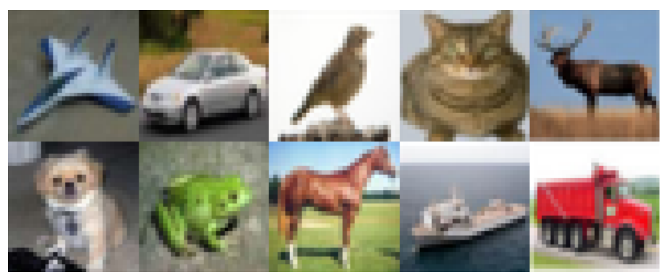

Fig. 8: Real images from each of the CIFAR-10 categories (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck) which were found to be most recognizable by DEEP-CNET, winner of the 2015 Kaggle CIFAR-10 Competiton.

IV. CIFAR-10 Experiments

Our second set of experiments used color images from the CIFAR-10 dataset [10] (see Fig. 8). The HERCL artist is required to output three values for each pixel, in order to generate a color image, and the architecture of the LeNet-5 critic is slightly modified to handle three inputs for each pixel.

A. Single-Cell Artists trained against All CIFAR-10 images

Three runs each were initially performed for Continue+All and Reset+All, using the full set of CIFAR-10 images. On average, one new image was produced every 2 minutes on a standard desktop machine. But, occasionally, the evolution took up to an hour because no artist was able to fool the critic into assigning a cost below the specified threshold.

Fig. 9: Every 50th image from a sequence of 500 images generated using LeNet-5 CNN and CIFAR-10 (all categories) with Continue+All (upper) and Reset+All with limited evolution (lower).

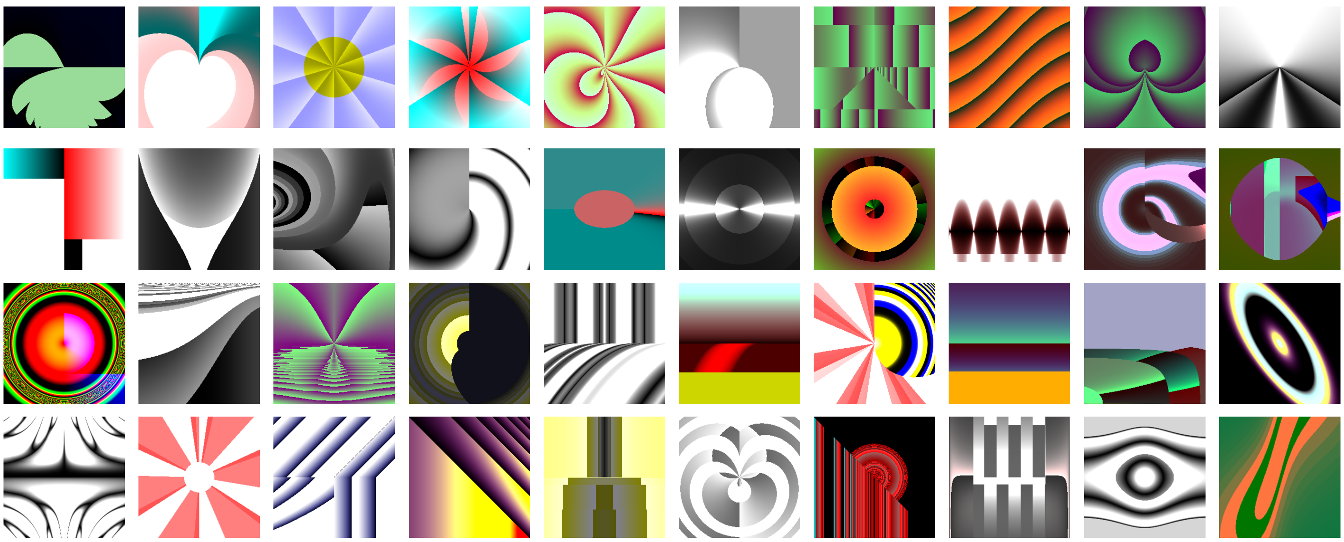

Fig. 10: Selected images with CIFAR-10 and LeNet-5, from two runs using Continue+All (top row) and five runs using Reset+All (lower three rows).

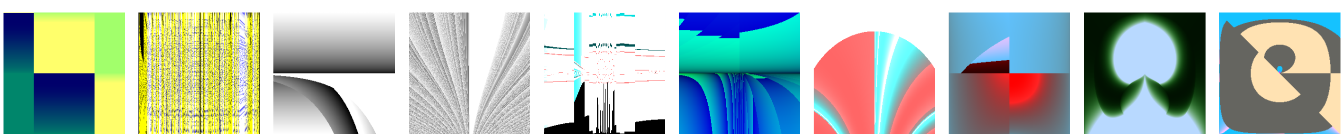

Fig. 11: Selected images using 4-cell HERCL programs, Reset+All, CIFAR-10 (all categories) and Lenet-5.

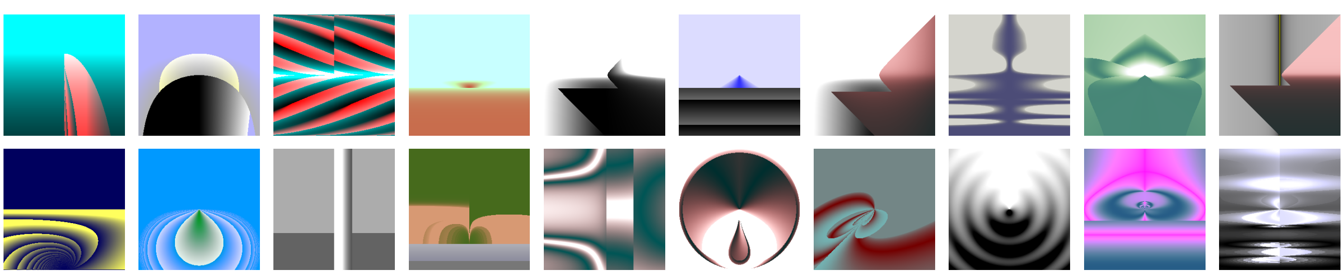

Fig. 12: Selected images using 4-cell HERCL programs, Reset+All and only the Ship images from CIFAR-10, with LeNet-5 (first 3 images in top row) and All Convolutional Net (remaining 17 images).

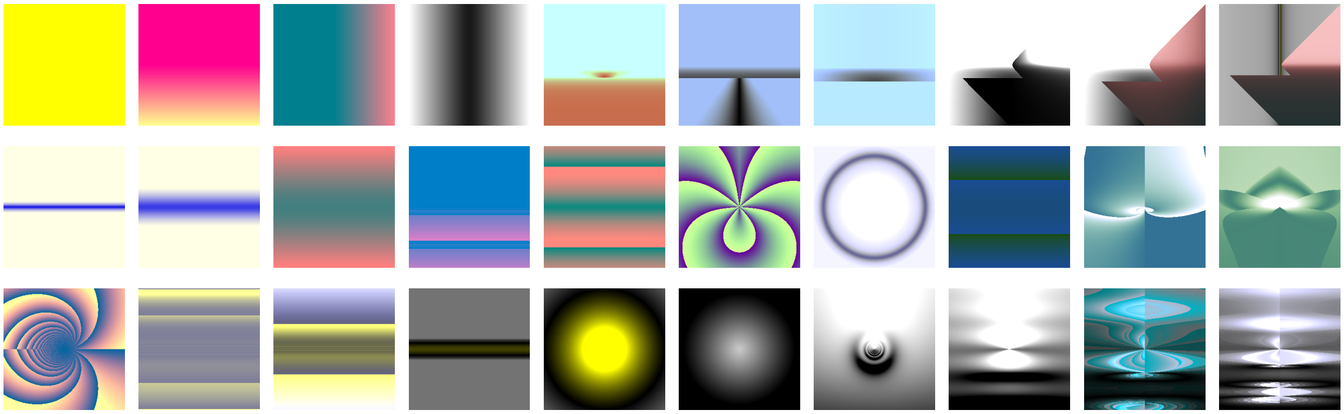

Fig. 13: Partial ancestry of three images from the 4-cell, Reset+All, Ships only, All Convolutional evolutionary run.

Therefore, we tried two additional runs of Reset+All in which the evolution was cut off if it ran for too many evaluations (200 000) without achieving the target cost.

We have extracted every 50th image from two of these runs in order to show the typical distribution of images (Fig. 9) and also selected a number of other images from all the runs, to give an idea of the variety of images generated (Fig. 10). In each case, images are arranged left-to-right according to where in the run they were generated. In a typical run, the first 20 or so images are just a single solid color, while the next 30 or so images consist of colors and shades broken up by horizontal and/or vertical lines – which somewhat resemble national flags, or colored patterns in the style of Mondrian or Paul Klee. Subsequent images get progressively more complex, often relying on fractal self-symmetry [1]. Some images remind us of everyday objects like a wine glass, wheel, spoon, disk, Christmas ornament or a row of popsicles. Others appear as idealized landscapes or seascapes, or art deco style buildings or patterns. In cases where the maximum number of generations has been exceeded without achieving the target cost, the image is sometimes simply a diagonal striped pattern.

B. 4-Cell Artists trained against All CIFAR-10 images

Our next experiment employed the same framework, but with the size of the HERCL programs increased to 4 "cells", which effectively allows subroutine calls to be included in the code. Selected images from this run are shown in Fig. 11. Compared to the single-cell images, we see that the 4-cell images are generally of greater complexity, and more likely to be built up as a composite of different sub-images.

C. 4-Cell Artists trained against Ship images

It is possible that the modified LeNet architecture for the critic may be inadequate, since similar architectures have been known to fail on the task of classifying the CIFAR-10 images. Also, the CIFAR-10 database includes a mix of animals and vehicles, which might make the job harder for the critic.

For our final set of experiments, in order to see whether the system could produce more realistic images, we restricted the set of real images to just the "Ship" category within the CIFAR-10 dataset. For one of these runs, we also used a modified All-Convolutional network for the critic (Table II), which takes longer to run but is more powerful. For this run, we additionally extended the running time to 1000 images (compared to 500 images for all the previous experiments).

Selected images from the two runs using only Ship images are shown in Fig. 12. In many cases, they do indeed resemble ships, nautical objects or seascapes – often with a small ship-like object in the middle or on the horizon.

D. Geneology of Images

Each image is the result of a mini-evolution, in which the code for all previously generated images is available as raw genetic material in the library. We can define the *primary parent* of an image as the previous image in the sequence whose code is most similar to that of the current image, according to the Levenshtein Edit Distance [12]. This definition of primary parent allows the images to be arranged in a family tree. The partial ancestry of three images from the All Convolutional, Ship-only experiment are shown in Fig. 13. We see that the complexity generally increases from parent to child, and that genetically related images do exhibit some similarity in style and content (although they may not be close in terms of raw Euclidean distance).

V. Discussion, Conclusion and Future Work

We have shown that a fully autonomous system with no human intervention can generate realistic digits, as well as geometric shapes and patterns which stimulate our visual system and remind us, in a way, of real objects, landscapes and designs.

Some may criticize our approach on the grounds that the images are on a small scale and lack the gravitas, contextual grounding or social and political commentary of "serious" art. However, our system is effectively solving a constrained optimization problem: namely, to generate images which have low algorithmic complexity, but are sufficiently suggestive of real-world images as to fool a trained critic with an architecture loosely modeled on the human visual system. It could be argued that this same characterization also applies (at least in part) to abstract art of the early 20th century.

Recent trends in networking and social media have seen a shift – in text, video and photography – away from long-form works and in favor of ephemeral pieces, broken into bite-sized chunks. Our system, and others of similar design, could perhaps represent a step in the same direction for visual art, by producing micro-artworks or "snack art" which can be enjoyed in the moment and then saved or discarded.

Another promising approach is the combination of automated and human-induced fitness evaluation [4, 19]. Our current (fully autonomous) system produces images of varying quality. While some are compelling, others may be unattractive or fail to meet the aesthetic desires of a particular human observer. In future work, we aim to bring some element of interactive evolution into this framework – by creating a system where images are generated autonomously in the background, but the human is invited to intervene from time to time to select their preferred images, and thus guide the evolution in a more aesthetically pleasing direction. This kind of hybrid approach may help to bring evolutionary art to a higher level, by saving human effort but at the same time giving the human enough control over the system to produce images which are in a desired style but still with the capacity to surprise.

ACKNOWLEDGMENT

This research was undertaken with the support of Akin.com, as well as resources and services, provided through Intersect and NCMAS, from the National Computational Infrastructure (NCI), which is supported by the Australian Government.

REFERENCES

[1] Barnsley, M., 1993. *Fractal Image Compression*, AK Peters, Natick, MA.
[2] Blair, A., 2013. "Learning the Caesar and Vigenere Cipher by hierarchical evolutionary re-combination", IEEE Congress on Evolutionary Computation, 605-612.
[3] Dawkins, R., 1986. *The Blind Watchmaker: why the evidence of evolution reveals a world without design*, Norton, New York.
[4] Funes, P., E. Sklar, H. Juillé & J. Pollack, 1998. "Animal-animat coevolution: Using the animal population as fitness function", From Animals to Animats (SAB-5), 525-533.
[5] Galanter, P., 2012. "Computational aesthetic evaluation: past and future", In *Computers and Creativity*, Springer, 255-293.
[6] Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville & Y. Bengio, 2014. "Generative adversarial nets", Advances in Neural Information Processing Systems, 2672-2680.
[7] Greenfield, G. & P. Machado, 2009. "Simulating Artist and Critic Dynamics - An Agent-based Application of an Evolutionary Art System", Proc. Int'l Joint Conf. on Computational Intelligence (IJCCI), Funchal, Madeira, Portugal, 190-197.
[8] Kingma, D.P. & J.L. Ba, 2015. "Adam: a method for stochastic optimization", International Conference on Learning Representations, 1-15.
[9] Kowaliw, T., A. Dorin & J. McCormack, 2012. "Promoting creative design in interactive evolutionary computation", *IEEE Transactions on Evolutionary Computation* 16(4), 523-536.
[10] Krizhevsky, A., 2009. Learning multiple layers of features from tiny images, Master's Thesis, Computer Science, University of Toronto.
[11] LeCun, Y., L. Bottou, Y. Bengio & P. Haffner, 1998. "Gradient-based learning applied to document recognition", Proceedings of the IEEE 86(11), 2278-2323.
[12] Levenshtein, V.I., 1966. "Binary codes capable of correcting deletions, insertions, and reversals", *Soviet Physics Doklady* 10, 707-710.
[13] Li, Y. & C. Hu, 2010. "Aesthetic learning in an interactive evolutionary art system", *Applications of Evolutionary Computation*, Springer, 301-310.
[14] Machado, P., J. Romero & B. Manaris, 2008. "Experiments in computational aesthetics: An Iterative Approach to Stylistic Change in Evolutionary Art", *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*, Springer, 381-415.
[15] Nguyen, A., J. Yosinski & J. Clune, 2015. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images", Proc. IEEE Conf. Computer Vision and Pattern Recognition, 427-436.
[16] Nguyen, A., J. Clune, Y. Bengio, A. Dosovitskiy & J. Yosinski, 2017. "Plug & play generative networks: Conditional iterative generation of images in latent space", IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 3510-3520.
[17] Pollack, J.B. & A. Blair, 1997. "Co-evolution in the successful learning of backgammon strategy", *Machine Learning* 32(3), 225-40.
[18] Secretan, J., N. Beato, D.B. D'Ambrosio, A. Rodriguez, A. Campbell, J.T. Folsom-Kovarik & K.O. Stanley, 2011. "Picbreeder: A case study in collaborative evolutionary exploration of design space", *Evolutionary Computation* 19(3), 373-403.
[19] Rooke, S., 2002. "Eons of genetically evolved algorithmic images", in: *Creative evolutionary systems*, P.J. Bentley & D.W. Corne, Eds., Morgan Kauffmann, pp. 339-365.
[20] Saunders, R. & J.S. Gero, 2001. "Artificial creativity: A synthetic approach to the study of creative behaviour", Computational and Cognitive Models of Creative Design V, Key Centre of Design Computing and Cognition, University of Sydney, 113-139.
[21] Sims, K., 1991. "Artificial evolution for computer graphics", *ACM Computer Graphics* 25(4), 319-328.
[22] Springenberg, J.T., A. Dosovitskiy, T. Brox & M. Riedmiller, 2014. "Striving for Simplicity: The All Convolutional Net", ICLR Workshop. [Online]. Available: http://arxiv.org/abs/1412.6806
[23] Vickers, D., J. Soderlund & A. Blair, 2017. "Co-Evolving Line Drawings with Hierarchical Evolution", Aust. Conf. on Artificial Life and Computational Intelligence (ACALCI), LNAI 10142, 39-49.