Evolutionary Reinforcement Learning Approaches to Computational Creativity: The Abstract Portrait Painter

Hussain, Alkhorsan 170365952

School of Electronic Engineering and Computer Science, Queen Mary University London, UK

ec211157@qmul.ac.uk

Abstract. Computational creativity has been applied in many domains that requires the generation of content, procedural generation via rule-based mathematical algorithms have been applied to automate game design and generative models via deep learning-based approaches have been used to generate synthetic data in many forms including images, video and audio. One way computational creativity can be applied in the art domain is to use evolutionary computation inspired by biological evolution to generate artistic digital representations of people by iteratively optimizing the placement of vector polygons on a canvas.

In this work, an evolutionary reinforcement learning algorithm is explored as an alternative to traditional oil paintings, photographs and handmade artefacts by generating a population of solutions drawn from a search distribution and using policy optimization via stochastic gradient ascent to improve the fitness of future generations. Several optimization algorithms and fitness metrics were studied for comparison. The experiments show that the Adam optimizer combined with the per-pixel loss function generates the best results and converges faster than other configurations.

1 Introduction

During the late 19th century, the rise of modernism has influenced art cultures to explore new forms of artistic expression. Series of experimental concepts have led society to move on from traditional rules of perspective and instead use more abstract representations. For instance, the revolutionary Cubism art movement popularized by influential artists such as Picasso transformed the way art is perceived by breaking up objects and reassembling them in abstracted form [1]. The artist Kandinsky who was a pioneer of abstracted art was famous for using primitive styles in his work consisting of geometric representations to achieve abstractionism that expresses the inner feelings of the artist [2]. More recently, minimalist art was introduced as a reaction against abstractionism to explore other forms of expressionism [3]. Borrowing philosophies from minimalist art, computer art also became prevalent during the late 1990s using algorithms to generate artefacts in many forms and similarly evolutionary art – a branch

of generative art in which the human does not construct the work but guides a system to produce artefacts in an iterative process of selection and mutation to achieve a satisfactory output has also become popular recently [4]. The most common approach to evolutionary art is to use genetic algorithms to represent images using basic polygons to achieve an abstract representation of the target image. This method became the standard approach to evolutionary art within the computational creativity community and has also been served as a basis for several interesting extensions [5] [6] [7]. The contribution of this work is to propose an alternative to current evolutionary approaches by borrowing ideas from the reinforcement learning domain to guide the evolutionary process and to also demonstrate how this approach can be used as the basis for other generative systems. Modern art heavily relies on expressionism and revealing information about its creator. For this reason, the work in this paper looks at how the system might be able to express itself to its audience using the notion of framing in computational creativity theory which aims to provide a narrative on the motivation behind the systems artefacts [8].

2 Background

Several works [5] [6] have utilized genetic algorithms as a tool to approximate a target image by using primitive styles such as polygons, circles and lines. The placement of objects is chosen by iteratively evolving a population of genotypes and evaluating their fitness relative to a user-defined fitness function. The most common type of fitness metric used when approximating an image is to compute the pixel-wise difference between the solutions and the phenotype. Subsequently, the best genotypes are selected and occasionally mutated with a pre-defined probability. Furthermore, alternative techniques have been explored such as the Plant Propagation Algorithm (PPA) to optimize solutions, PPA assigns normalized fitness values to each individual in the current population where each individual produces a number of offspring proportional to its fitness and is mutated inversely proportional to its normalized relative fitness. This dynamic algorithm allows fitter individuals to produce more offspring with less mutation and less fit individuals to produce less offspring with higher mutations [7]. In addition to these works stochastic hill-climbing and simulated annealing methods have also been explored in [9] [10]. The domain of reinforcement learning is concerned with training an agent to take optimal actions in order to maximize the notion of cumulative reward by learning a policy with respect to its environment. RL techniques are very popular in situations that can be modelled as Markov decision processes (MDP). Borrowing ideas from reinforcement learning and evolutionary computation, distribution-based search algorithms aim to approximate the gradient of the reward with respect to the policy's parameters by sampling a population of solutions from a search distribution and optimizing the policy using a gradient-based technique. Although, these methods are mainly used in RL research, they are considered general-purpose and can be applied to many gradient-based optimization problems [11].

The concept of maximizing a reward in reinforcement learning is crucial in understanding the motivation behind an agent's actions. The contribution of this work is to provide a unique idea of framing within the context of computational creativity by arguing that the reward-seeking behavior of a system can be considered the way in which it expresses itself through the nature of its work.

3 Methodology

Policy gradients with parameter-based exploration

In this work, an evolutionary reinforcement learning technique is used to generate the artefacts. The experiments use policy gradients with parameter-based exploration (PGPE) in order to guide the search. In each iteration of the PGPE algorithm, a population of solutions are constructed by sampling neighboring solutions from a Gaussian distribution around the current solution. The gradient of the policy is then approximated by computing the weighted average of all the fitness gains in each direction and then updating the standard deviation vector. The gradients are used to update the current solution and standard deviation. PGPE is a maximizing algorithm and therefore uses stochastic gradient ascent. Algorithm 1 shows the steps taken per iteration.

Algorithm 1 PGPE algorithm

Hyperparameters:

Number of iteration $\mathbf{n}$
Population size $\boldsymbol{\lambda}$
Initial solution $\boldsymbol{x_0}$
Initial standard deviation $\boldsymbol{\sigma_0}$
Learning rate Ω
Adaptive optimizer $\in \{\mathbf{Adam}, \mathbf{ClipUp}\}$

for $\mathrm{k} = 0, 1, 2, \ldots \mathrm{n}$ **do**

1. Construct population of search directions
$$\delta_i \sim (N(0, \mathrm{I}) \cdot \sigma_k) \qquad \mathrm{i} \in \{1, 2, \ldots, \lambda/2\}$$

2. Estimate the gradient for updating the current solution $\boldsymbol{x_k}$

3. Estimate the gradient for updating the standard deviation vector $\boldsymbol{\sigma}$

4. Update the current solution and standard deviation
$$\mathrm{x_{k+1}} \leftarrow \mathrm{x_k} + \mathrm{AdaptiveOptimizer}(\nabla_{\mathrm{x_k}})$$
$$\sigma_{\mathrm{k+1}} \leftarrow \sigma_{\mathrm{k}} + \Omega \cdot \nabla_{\sigma_{\mathrm{k}}}$$

end for

Fitness function

The fitness of each generation is computed with respect to a fitness metric. Most image approximation techniques use the per-pixel loss between two images as fitness by computing the total absolute errors between each pixel. However, the perceptual pixel loss can also be used to capture more high-level representations. The resulting loss of each generation is then fed back into the algorithm to update the parameters.

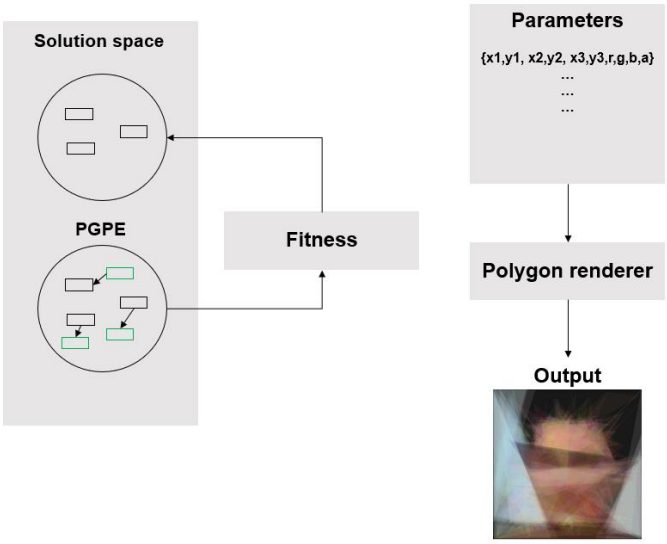
Per-pixel loss:

$$Loss(x, y) = \frac{1}{n}\sum[x_i - y_i]^2$$

Polygon renderer

The polygon renderer draws the polygons on an initially empty canvas of the same height and width of the target. It takes as input user specified parameters such as the height, width of image, number of triangles, alpha scale and coordinate scale. The coordinates and shape of the polygons are derived from the parameters of the PGPE algorithm. After the parameters are rendered, the corresponding image is evaluated using the fitness function in order to update the current solution.

System architecture



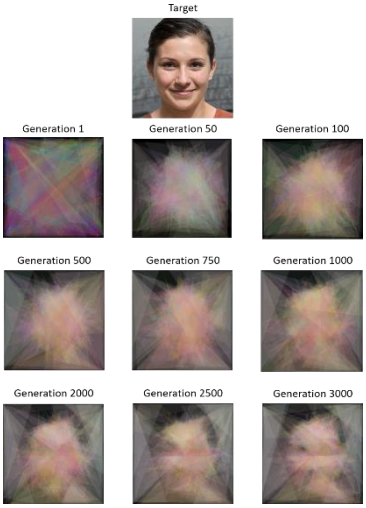
4 Experiments and Results

Each experiment was run for 3000 iterations to assess the output on different configurations. In total, 6 experiments were run with one specific hyperparameter modified to check whether it improves the quality of the artefacts. Table 1. shows the configuration used for each experiment.

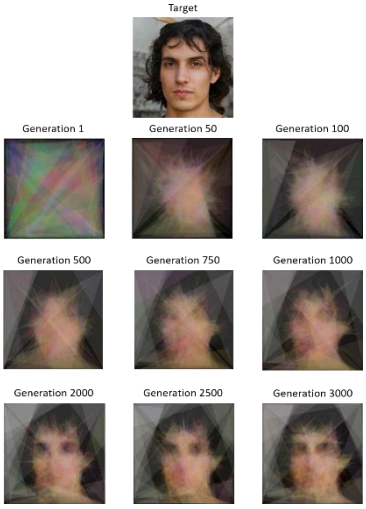
Table 1. Configuration of different hyperparameters

Exp	Optimizer	Population size	Loss function	Iterations
Exp 1	Adam	128	Per-pixel loss	3000
Exp 2	Adam	256	Per-pixel loss	3000
Exp 3	Adam	512	Per-pixel loss	3000
Exp 4	ClipUp	128	Per-pixel loss	3000
Exp 5	ClipUp	256	Per-pixel loss	3000
Exp 6	ClipUp	512	Per-pixel loss	3000

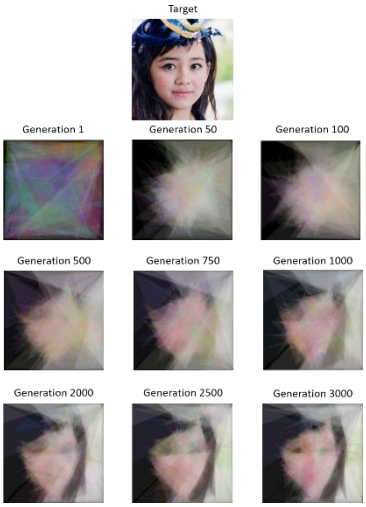
The loss from each experiment was recorded to compare how well the system is approximating the target image. The results show that Exp 3 using the Adam optimizer with a population size of 512 gave the lowest loss after 3000 iterations. However, the results from Exp 1 and Exp 2 using the Adam optimizer with a population size of 128 and 256 respectively compared very similar to Exp 3 in terms of its loss over training which suggests that increasing the population size had little effect on the output. The main distinction between the results from the experiments was the type of optimizer used. The ClipUp optimizer used on Exp 4, Exp 5 and Exp 6 performed worse than when using the Adam optimizer. It is also worth noting that during the first 500 iterations, the loss decreases gradually then converges suggesting that higher iterations often had little effect on the output and solutions generated. Figure 1. shows the output from each configuration and Figure 2. shows the loss function from each experiment during training.



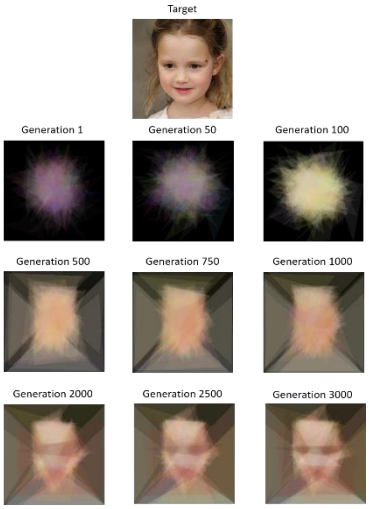
Exp 1. Adam optimizer, pop-size=128, 3000 iterations.



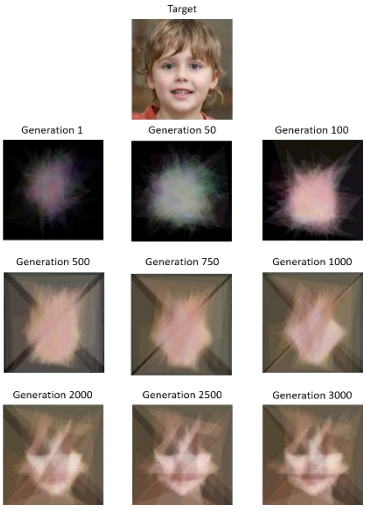
Exp 2. Adam optimizer, pop-size=256, 3000 iterations.



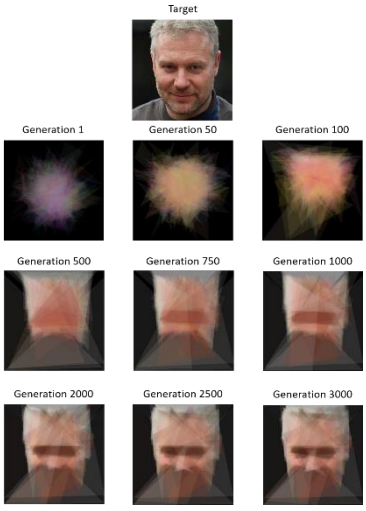
Exp 3. Adam optimizer, pop-size=512, 3000 iterations.



Exp 4. ClipUp optimizer, pop-size=128, 3000 iterations.



Exp 5. ClipUp optimizer, pop-size=256, 3000 iterations.



Exp 6. ClipUp optimizer, pop-size=512, 3000 iterations.

Figure 1. Examples – experiments 1-6

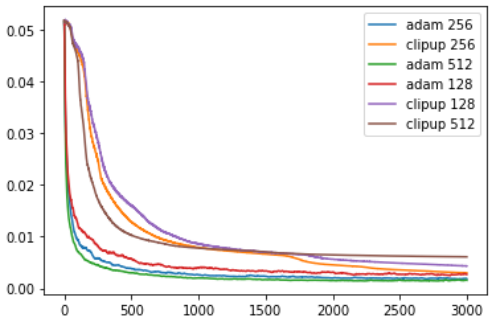


Figure 2. Loss curve – experiments 1-6

It is also worth mentioning that the quality of the artefacts generated by the system are not always optimal – particularly when a selection of hyperparameters are not chosen carefully. During experimentation this was found to be an issue mostly with the population size and number of parameters to render. The best results were generated by using a population size between 128 and 256 with a range of 3000 – 5000 parameters. Lower ranges often lead to bad results as shown in Figure 3. and higher ranges significantly increased training time with no improvement on the quality. The importance of showing where the system can fail is due to the fact that the selection of the right parameters is attributed to the person who selects them and not the creativity of the system itself as it has no way of evaluating the combination of hyperparameters that lead to the best results – a metric that is commonly used in evaluating creative systems in computational creativity theory [13].

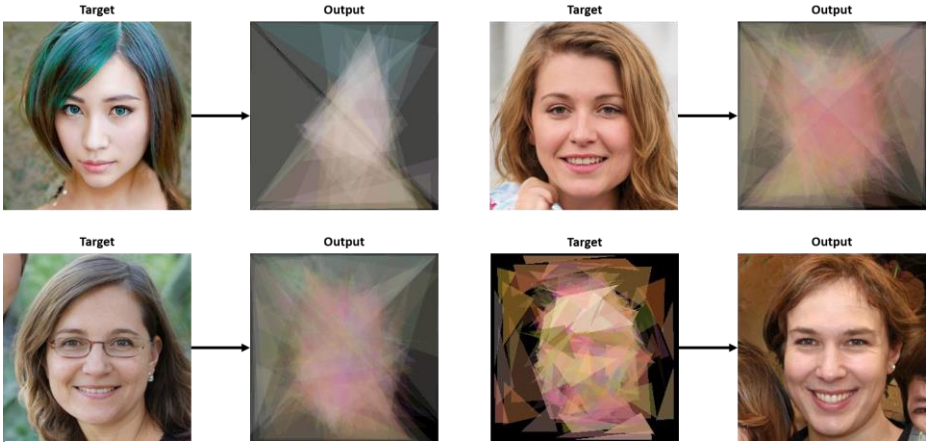


Figure 3. Examples using non-optimal hyperparameters

5 Discussion

The results obtained in the experiments show that ideas from reinforcement learning and evolutionary computation can be combined to generate interesting artefacts. Specifically, the goal-oriented nature of reinforcement learning techniques as well as the novelty and diversity of evolutionary computation can serve as a good basis for computational creativity. Furthermore, the artefacts generated from the experiments show a high-level representation of the subject which resembles a form of abstract art using computational methods. Some ideas that were explored was to use some notion of reward as the basis for the fitness function instead of the per-pixel loss but these attempts were unsuccessful due to the complexity of modelling a suitable reward function. Although, the per-pixel loss function could also be considered as a simple reward mechanism. Currently, the system attempts to draw images from an external generative system found online [12] which are artificial samples of faces. One attempt was to have the system generate its own data and use samples from it to render portraits however, this requires building another model as input to the main system which was time-consuming given the time available and the underlying difficulty in collecting enough data and therefore was left as potential for future research.

6 Conclusions and Future Work

The main contributions of this work are to explore how a reward mechanism in reinforcement learning can serve as the motivation behind a systems generative output. By modelling a sophisticated reward function, the system can generate artefacts with respect to maximizing its notion of reward. So far, this work has explored a system that is able to generate an approximate representation of an image using techniques from abstract art. Some potential future work might explore how to hand over more responsibility to the system to generate artefacts it considers impressive with respect to its own judgement. One way to achieve this is by having a suitable reward mechanism that gives it feedback on the quality of its work where good artefacts are rewarded and bad artefacts are penalized. This feedback could be from a human audience or alternatively from a prior probability distribution. For instance, by giving the system a probability distribution of a diverse range of artefacts it could be rewarded positively based on how similar its artefacts are to the prior probability. After this, the system can even be made more independent by generating artefacts similar in nature to its own previous artefacts, giving the system more freedom to express its creativity.

References

[1] Rewald, S. 2004. Cubism.

[2] Stella, P. 2004. Abstract Expressionism.

[3] Strickland, E. 1993. Minimalism: Origins.

[4] Lewis, M. 2008. Evolutionary Visual Art and Design.

[5] Berg, J.; Berggren, N.G.A.; Borgeterien, S.A.; Jahren, C.R.A.; Sajid, A.; and Nichele, S. 2019. Evolved Art with Transparent, Overlapping, and Geometric Shapes.

[6] Johansson, R. 2008. Genetic Programming: Evolution of Mona Lisa.

[7] Paauw, M., and Berg, D.V.D. 2019. Paintings, Polygons and Plant Propagation.

[8] Cook, M.; Colton, S.; Pease, A.; and Llano, M.T. 2019. Framing In Computational Creativity – A Survey And Taxonomy.

[9] Dahmani, R.; Boogmans, S.; Meijs, A.; and Berg, D.V.D. 2020. Paintings-from-Polygons: Simulated Annealing.

[10] Berenguel, R. 2010. Approximating images with randomly placed translucent triangles.

[11] Toklu, N.E.; Liskowski, P.; and Srivastava, R.K. 2020. ClipUp: A Simple and Powerful Optimizer for Distribution-based Policy Evolution.

[12] https://thispersondoesnotexist.com

[13] Colton, S., and Wiggins, G.A. 2012. Computational Creativity: The Final Frontier?