TGPGAN - Towards Expression-based Generative Adversarial Networks

Francisco Baeta
University of Coimbra, CISUC, DEI
Coimbra, Portugal
fjrbaeta@student.dei.uc.pt

João Correia
University of Coimbra, CISUC, DEI
Coimbra, Portugal
jncor@dei.uc.pt

Tiago Martins
University of Coimbra, CISUC, DEI
Coimbra, Portugal
tiagofm@dei.uc.pt

Penousal Machado
University of Coimbra, CISUC, DEI
Coimbra, Portugal
machado@dei.uc.pt

ABSTRACT

Before the advent of Generative Adversarial Networks (GANs), Evolutionary Computation approaches made up the majority of the state of the art for image generation. Adversarial models employing Deep Convolutional Neural Networks better fitted for GPU computing have been at this point more efficient. Nevertheless, motivated by recent successes in GPU-accelerated Genetic Programming (GP) and given the disposition of expression-based solutions towards image evolution, we believe in the prospect of using symbolic expressions as generators for GANs, instead of neural networks. In this paper, we propose a novel GAN model called TGPGAN, where the traditional convolutional generator network is replaced with a GP approach. The generator iteratively evolves a population of expressions that are then passed to the discriminator module along with real images for backpropagation. Our experimental results show that it is possible to achieve comparable results to a typical Deep Convolutional GAN while benefiting from the flexibility enabled by an expression-based genotype. Moreover, this work serves as a proof of concept for the evolution of symbolic expressions within adversarial models.

CCS CONCEPTS

• **Computing methodologies** → **Generative and developmental approaches**; **Genetic programming**;

KEYWORDS

Expression-based evolution, Genetic Programming, Generative Adversarial Networks, TGPGAN

ACM Reference Format:
Francisco Baeta, João Correia, Tiago Martins, and Penousal Machado. 2022. TGPGAN - Towards Expression-based Generative Adversarial Networks. 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/10.1145/3520304.3529064

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-1-4503-9268-6/22/07.
https://doi.org/10.1145/3520304.3529064

1 INTRODUCTION

In the seminal 2014 paper, Goodfellow et al. introduced Generative Adversarial Networks (GANs) [11], a new approach that was quick to revolutionize the training dynamics of generative models. Since its inception, the GAN model has taken traction comparatively to other generative models, with impressive results in several domains, of which image generation proved to be one of the most successful. In essence, GANs employ an adversarial training model consisting of a generator and a discriminator, which are mutually trained by optimize opposing metrics within in a zero-sum game.

However, despite their self-evident generative capabilities, GANs are still plagued by training issues such as mode collapse, instability, amongst other problems that compromise their generative performance [10]. For this reason, several variants based upon the traditional GAN model were suggested in attempts of augmenting the robustness of the learning process [1, 20]. Most of the proposed approaches were, nonetheless, either met with partial success, solved the problem at the cost of extra computational resources or took advantage of otherwise unrealistic datasets that are either too large or too complete to represent real-world scenarios. We believe that a possible solution to such problems may lie in a field not so strange to generative modelling.

Before the proposal of GANs, Evolutionary Computation (EC) was, arguably, the most commonly used approach in the context of image generation [9]. In particular, expression-based evolution techniques, such as Genetic Programming (GP), constituted efficient means of representing and evolving images [13, 16]. Nevertheless, these approaches tended to be computationally expensive and largely fell into disuse in favor of image-specific GAN variations such as Deep Convolutional Generative Adversarial Networks (DCGANs) that take advantage of the parallelization capabilities of modern hardware (such as GPUs and TPUs).

Recently, a substantial body of research has focused on aiding the traditional adversarial pipeline using Evolutionary Machine Learning (EML) approaches that aim to combine the discriminative power of ML with the representation flexibility provided by EC. This way, motivated by recent successes in accelerating domain evaluation in GP using GPUs. we believe that EC can be reconsidered as a viable option to tackle the aforementioned training issues of GANs without the excessive computational burden. Namely, in this work, we implement and test a model that serves as a proof of concept for the "raw" evolution of symbolic expressions within

an adversarial model; an idea that is somewhat unexplored in literature. The model we propose, named TGPGAN, implements a conventional DCGAN with a standard convolutional discriminator but with an evolutionary GP run serving as the generator module.

The outline for this paper is as follows. The following section provides an analysis of the literature review of the interaction between EC and adversarial models in the image domain. Subsequently, a description of our model is presented along with relevant integration efforts. The experimentation section follows with a documentation of the preliminary testing carried out with the model and corresponding results. Finally, the main conclusion points from this work are highlighted, along with some problems to tackle in future work.

2 LITERATURE REVIEW

Perhaps the most intuitive way to couple EC techniques with ML algorithms is to directly evolve a population of ML solutions. To stabilize GAN training and improve generative performance, Wang et al. proposed Evolutionary GANs (EGANs) [19], which saw the evolution of populations of generators that adapt to the environment through normal selection and reproduction. Costa et al. then extended the EGAN model by integrating neuroevolution and coevolution, aimed at optimizing deep learning architectures through the evolutionary process, introducing CoEGANs [6].

Early efforts in expression-based image generation were performed by Karl Sims using symbolic expressions as genotypes for the artificial evolution of generative art [9] with representations reminiscent of Compositional Pattern Producing Networks (CPPNs) [16]. Another popular approach is The NeuroEvolution of Augmented Topologies (NEAT), which focuses on evolving both the architecture and the weights of a neural network thus enabling the evolution of increasingly more complex solutions using genetic encodings[18]. In his seminal work, Stanley combined the proposed CPPN approach with NEAT, introducing the aptly named CPPN-NEAT model and demonstrating the power of neuroevolution when applied to generative models [17]. Ha then applied the CPPN-NEAT framework to generative art by modifying the model to accept a randomly initialized random vector z in order to allow for morphing images [12]. Ha also introduced the incorporation of the CPPN architecture with the adversarial pipeline using a Variational Autoencoder (VAE) GAN, aided by the standard backpropagation through the generator network. Nevertheless, the model trained on the CIFAR dataset demonstrated the limitations of this approach in generating a truthful representation of rich images. Later, Metz and Gulrajani [15], extended upon the proposed model by using a DCGAN architecture and training different discriminators for different datasets. Notwithstanding the impressive generative performance, the presented model was "unable to reach the same visual quality of existing convolution transposed models". More recent work by Ekern et al. further revealed the effectiveness of CPPN-GAN models [8]. However, there is still no study aimed at incorporating GP instead of CPPNs within the generator of GANs. Nonetheless, because GP retains, at its core, the same flexibility of representation as CPPNs, this research path seems motivating.

Lastly, one of the core challenges of coupling GANs with the expression-based paradigm lies in the fact that isolated solutions are generated instead of a latent space, which is characteristic of GANs and other generative approaches. Alternatives to mapping individual solutions into an organized latent space include the addition of an archive that stores solutions according to their fitness and novelty [5, 14], local embeddings [7], manifold learning algorithms [4] or VAEs that trains an additional network to learn image encodings.

3 TGPGAN

This section provides a description of TGPGAN. We start by introducing TensorGP, the GP engine used in our model, and follow this by presenting some of the interactions between the different components, as well as the challenges faced during the process of integrating expression evolution with a DCGAN pipeline.

TensorGP is the engine that powers the generator component of TGPGAN. It was developed to address arguably the biggest drawback to GP: its computational demands. Despite requiring the evaluation of each individual for all fitness cases, the expression to evaluate remains constant, making GP highly parallelizable. Besides, the mechanism of evolution makes it so that highly fit sub-expressions proliferate throughout generations, meaning that we can save these intermediated results to avoid recalculation. TensorGP applies both the parallelization of the GP operations and the caching of intermediate fitness results [2, 3]. The process of genotype to phenotype translation in TensorGP can be divided into three steps: transform the initial expression (the genotype) into a tree graph, convert the tree graph into a Directed Acyclic Graph (DAG) to avoid the recalculation of identical nodes, and finally, traversing the DAG to produce a tensor/image (the phenotype).

As mentioned, the model that we propose in this work is a variation of a conventional DCGAN where, instead of a convolutional network for the generator, we have a population of symbolic expressions being evolved at each training step. In this paradigm, the discriminator module remains a Convolutional Neural Network (CNN), which is trained by standard backpropagation using both real and generated images. For the current iteration of the model, each training step can be defined by the pseudocode in Algorithm 1. The source code for this project is publicly available on GitHub. [1]

Algorithm 1: Main training loop of the model.

```
foreach training step do
    Generate batch from GP run with n generations, and k
      best individuals from last step if applicable;
    Update solution archive;
    Get real batch from dataset;
    Train discriminator with real batch;
    Train discriminator with generated batch;
    Calculate losses;
```

The algorithm can be explained as follows. To generate the batch of fake images, a GP run consisting of n generations is performed using TensorGP. After performing n generations, the resulting individuals are then converted to their respective images phenotypes. In this evolutionary process, candidate solutions will be assessed by

[1] TGPGAN GitHub repository: https://github.com/AwardOfSky/TensorGP_DCGAN

a forward pass on the discriminator network. For the first training step, the initial GP population is randomly generated while in subsequent steps, a given portion of the best-fitted individuals is taken from the last population. Similarly to the standard generational elitism, this mechanism enables us to implement a certain degree of elitism for the generator across training steps. Next, another batch of images is retrieved from the original dataset. After this, the weights of the discriminator network are updated by passing in both the real and generated data samples for standard backpropagation training coupled with the Adam Optimizer. The described algorithm is then repeated for each training epoch.

To address the problem of generating an organized latent space, an archive of solutions was implemented to collect the best-fitted individuals found throughout the training process. The archive is updated on each training step where only the fittest individuals amongst the generated ones and the ones already in the archive are chosen. This ensures that individuals with lower fitness values are kicked out if better ones are produced by the generator.

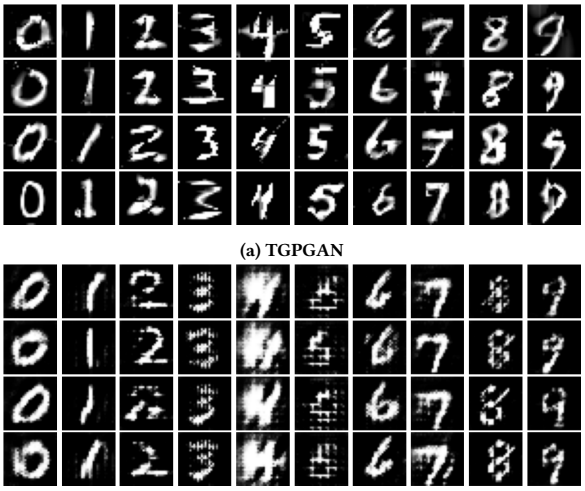
4 EXPERIMENTATION

This section defines the experimental setup used and provides an analysis of the results from training TGPGAN on the MNIST dataset. Despite many considering the MNIST dataset to be too simplistic of a classification task for most modern ML applications, the fact that the evolution of symbolic expressions in adversarial models resorting to GP is largely uncharted territory in literature warrants a more basic start as some validation is needed before moving onto more challenging tasks. In addition to the experimentation on our model, we include comparative results from a conventional DCGAN model, adapted from TensorFlow's website, which represents a quick baseline ideal for prototyping.[2]

Table 1: Parameterization of the TGPGAN model and GP run.

Parameter	Value
Generations	50
Population size	32
Elitism (Elite Size)	1
Meta-elitism	1
Tournament size	2
Mutation probability	0.3
Mutation operators	delete, insert, point, subtree
Crossover probability	0.8
Crossover operators	random sub-tree swap
Initial Depth [min, max]	[3, 6]
Allowed Depth [min, max]	[3, 14]
Generation method	RHH (population)
Function Set	add, sub, mul, div, abs min, max, neg, warp sign sqrt, pow, mdist, sin, cos, if

Concerning TGPGAN, the generator is a Genetic Programming run parameterized according to Table 1. As alluded to in the previous section, the "meta-elitism" parameter refers to the number of best-fitted individuals that are taken from the last population of the previous training step. Preliminary experimentation performed with the framework consisted in training the model on the entire dataset for 5 epochs. This task proved difficult as many digits as targets made the generated populations rather random in nature. For this reason, we decided to focus on the task of evolving each digit individually, keeping the same 5 epochs with a batch size of 32.



(a) TGPGAN

(b) DCGAN

Figure 1: Best-fitted individuals stored in TGPGAN's archive throughout the training process (top) versus generated throughout the training process for the DCGAN mode (bottom).

The results for the best-fitted populations after training are shown in Figure 1a. As demonstrated, the batches clearly start resembling digits after only 5 training epochs. It is also worth noting that the individuals manage to mimic the handwritten style of the digits present on the MNIST dataset by mixing various GP operators instead of producing simple geometric shapes. Additionally, we verify the variety of best-fitted solutions that were stored in the TGPGAN's archive throughout the training process. For instance, more complex digits such as 5's and 9's, which were shown to be difficult to generate in the previous experiment involving the whole dataset, proved evolvable at this stage. It is interesting to note of some of the individuals display the strike in the middle of digit 7 while others opt for a more simplistic approach but with different traces. This type of diversity is also made evident in some of the other digits.

For completeness, we provide the results of running the same setup on a typical DCGAN model specifically tailored for MNIST. The evolved model is in all aspects similar to the official TensorFlow implementation, except for the batch size, which was set to 32, alongside other minor necessary modifications to match the proposed pipeline. Akin to TGPGAN's archive, Figure 1b showcases, for every digit, some examples of the best-fitted individuals found throughout the training process for the DCGAN model. As we conclude, even though most of the imagery starts to share many resemblances to digits, it is still hard to make out certain instances being evolved such as for 5's and 4's. By querying an external classifier [1], we can get the percentage of correct classifications over the best-fitted populations on both models. The classifier used for this task is a CNN pre-trained on MNIST digits, reaching a test

[2]Discriminator adapted from: https://www.tensorflow.org/tutorials/generative/dcgan

accuracy of 99.25%. As shown, the average percentage of correct classifications across all digits for our model is 57% versus 47% for the DCGAN model.

Table 2: Percentage of correct digit classifications over the best-fitted population for all digits on both models (bold marks the highest percentage).

Digit	DCGAN	TGPGAN
0	**65.63%**	40.63%
1	68.75%	**96.88%**
2	43.75%	43.75%
3	96.88%	**100.00%**
4	**90.63%**	0.00%
5	12.50%	**87.50%**
6	0.00%	**18.75%**
7	93.75%	**100.00%**
8	0.00%	**12.50%**
9	0.00%	**9.38%**
Average	**47.19%**	**56.65%**

5 CONCLUSION AND FUTURE WORK

In this work, we propose a variation on the standard DCGAN model for image generation by replacing the generator module with a GP approach, while keeping the feedback of the discriminator to guide the evolutionary process. This approach is attractive because, along with the produced images, we also benefit from the corresponding symbolic expressions, allowing for an arbitrary level of resolution over the generated image phenotypes. The unaided evolution of expression-based solutions within adversarial models is an unexplored topic in ML literature. However, based on the success of EC approaches applied to the generation of images prior to the proposal of GANs, we believe this is mostly due to the computational cost imposed by expression evaluation. To alleviate the computational burden inherent to our proposed approach, we couple a standard convolutional discriminator implemented in Keras with TensorFlow and TensorGP, an engine specifically developed to accelerate the domain evaluation phase in GP.

Our experimentation focused on training TGPGAN on digits from the MNIST dataset. The choice of a more simplistic generation task lies in the need for validating our approach given the lack of exploration, to the best of our knowledge, in the related research. Results demonstrated that, after only 5 training epochs, the TGPGAN model managed to generate perceivable artifacts for every digit. Moreover, the developed model achieved a higher percentage of correct classifications on archived solutions across all digits when compared to a conventional DCGAN model. These successes were, however, achieved at the cost of more computational time to train the generator. More importantly, our developments should not be taken as a replacement for current adversarial models but rather as a proof of concept for the capabilities of expression-based evolution within the context of image generation.

We consider this the first step towards an expression-based GAN, and as such, some avenues should be considered for future work. As a priority, a more dynamic approach for organizing the latent space of expressions should be implemented. Feasible alternatives include using variational autoencoders and local embeddings alongside other types of manifold learning techniques. Likewise, reducing the computational cost of TGPGAN will be a requirement in bridging the gap between expression-based and GANs.

[1]Classifier taken from: https://github.com/kj7kunal/MNIST-Keras

ACKNOWLEDGMENTS

This research is partially funded by Fundação para a Ciência e Tecnologia (FCT), Portugal, under the grant SFRH/BD/08254/2021 and by European Social Fund, through the Regional Operational Program Centro 2020. We also thank the NVIDIA Corporation for the hardware granted to this research.

REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*. PMLR, 214–223.
[2] Francisco Baeta, João Correia, Tiago Martins, and Penousal Machado. 2021. Speed benchmarking of genetic programming frameworks. In *GECCO 2021 - Proceedings of the 2021 Genetic and Evolutionary Computation Conference*. ACM, to appear.
[3] Francisco Baeta, João Correia, Tiago Martins, and Penousal Machado. 2021. TensorGP — Genetic Programming Engine in TensorFlow. In *Applications of Evolutionary Computation – 24th International Conference, EvoApplications 2021*. Springer, 763–778.
[4] Lawrence Cayton. 2005. Algorithms for manifold learning. *Univ. of California at San Diego Tech. Rep* 12, 1-17 (2005), 1.
[5] João Correia, Penousal Machado, Juan Romero, Pedro Martins, and F Amílcar Cardoso. 2019. Breaking the Mould An Evolutionary Quest for Innovation Through Style Change. In *Computational Creativity*. Springer, 353–398.
[6] Victor Costa, Nuno Lourenço, João Correia, and Penousal Machado. 2019. COEGAN: Evaluating the coevolution effect in generative adversarial networks. In *Proceedings of the genetic and evolutionary computation conference*. 374–382.
[7] Victor Costa, Nuno Lourenço, João Correia, and Penousal Machado. 2021. Demonstrating the Evolution of GANs through t-SNE. *arXiv preprint arXiv:2102.00524* (2021).
[8] Erlend Gjesteland Ekern and Björn Gambäck. 2021. Interactive, Efficient and Creative Image Generation Using Compositional Pattern-Producing Networks. In *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*. Springer, 131–146.
[9] Philip Galanter. 2003. What is generative art? Complexity theory as a context for art theory. In *In GA2003–6th Generative Art Conference*. Citeseer.
[10] Ian Goodfellow. 2016. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160* (2016).
[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).
[12] David Ha. 2016. Generating Abstract Patterns with TensorFlow. *blog.otoro.net* (2016). https://blog.otoro.net/2016/03/25/generating-abstract-patterns-with-tensorflow/
[13] Penousal Machado and Amílcar Cardoso. 2002. All the truth about NEvAr. *Applied Intelligence* 16, 2 (2002), 101–118.
[14] Tiago Martins, João Correia, Ernesto Costa, and Penousal Machado. 2019. Evolving Stencils for Typefaces: Combining Machine Learning, User's Preferences and Novelty. *Complexity* 2019 (2019).
[15] Luke Metz and Ishaan Gulrajani. 2017. Compositional pattern producing GAN. In *NeurIPS Workshops*, Vol. 1.
[16] Karl Sims. 1991. Artificial evolution for computer graphics. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*. 319–328.
[17] Kenneth O Stanley. 2007. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines* 8, 2 (2007), 131–162.
[18] Kenneth O Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* 10, 2 (2002), 99–127.
[19] Chaoyue Wang, Chang Xu, Xin Yao, and Dacheng Tao. 2019. Evolutionary generative adversarial networks. *IEEE Transactions on Evolutionary Computation* 23, 6 (2019), 921–934.
[20] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.