The Generative Adversarial Random Neural Network

Will Serrano

The Generative Adversarial Random Neural Network

Will Serrano

The Bartlett
University College London

w.serrano@ucl.ac.uk

Abstract. Generative Adversarial Networks (GANs) have been proposed as a method to generate multiple replicas from an original version combining a Discriminator and a Generator. The main applications of GANs have been the casual generation of audio and video content. GANs, as a neural method that generates populations of individuals, have emulated genetic algorithms based on biologically inspired operators such as mutation, crossover and selection. This paper presents the Generative Adversarial Random Neural Network (RNN) with the same features and functionality as a GAN: an RNN Generator produces individuals mapped from a latent space while the GAN Discriminator evaluates them based on the true data distribution. The Generative Adversarial RNN has been evaluated against several input vectors with different dimensions. The presented results are successful: the learning objective of the RNN Generator creates replicas at low error whereas the RNN Discriminator learning target identifies unfit individuals.

Keywords: Generative Adversarial Networks; Random Neural Network.

1 Introduction

Generative Adversarial Networks [1] were proposed as a machine learning generative model for unsupervised learning. GANs consist basically of two neural networks which learning algorithms compete against each other: Discriminator (D) and Generator (G). The Discriminator (D) objective is to learn the properties of the individual based on true data in order to identify the Generator' created replicas. On the other side, the Generator (G) learning objective is to create data replicas by learning to map from a latent space to a data distribution of interest that deceive the Discriminator. The better an algorithm performs, the worse the other achieves and both learnings will reciprocally and dynamically try to improve their current performance. Generally, GANs suffer from four key drawbacks: 1) failure to converge between the discriminator and the generator. In addition, the difference in the dimensionality space between the source and the target may lead to unstable training. 2) model collapse where the generator only creates a reduced set of replicas. 3) diminished gradient due to an optimal discriminator feedbacks to the generator a vanished gradient inhibiting its learning. 4) overfitting when the generator does not create enough random replicas limiting

its creating capabilities to an error limit. Following this learning structure, GAN applications have been purely the generation of image [2] and video [3] content with ramifications to video games [4], art [5], fashion [6], advertising [7], and science [8]. GANs have also been used as variational autoencoders by learning a data distribution from a latent space [9]. Due to the random generation of replicas, quantum computing has also been applied to GANs [10,11]. Due to this generation of random examples from a common truth or set of rules, GANs have the potential to inspire or replace designers and artists in creative industries. GANs, as a neural method to generate populations of individuals, have emulated genetic algorithms based on biologically inspired operator such as mutation, crossover and selection.

1.1 Research proposal

This paper presents the Generative Adversarial Random Neural Network (RNN) with the same features and use cases as a GAN: an RNN Generator produces individuals mapped from a latent space while the GAN Discriminator evaluates them based on the true data distribution. The RNN [12-14] is a recurrent stochastic mathematical model of an interconnected neural network of neurons that exchange information via spiking signals. Excitatory spikes increase the potential of the receiving neuron whereas inhibitory spikes decrease it. Neuron fire spikes at random only when their potential is positive. The Random Neural Network represents more closely how signals are transmitted in many biological neural networks where they travel as spikes or impulses, rather than analogue signal levels. The RNN has already been applied as a process that gradually finds the ground truth (Generator) based on the direct user feedback (Discriminator) although not in direct adversarial learning. The iterative search and adaption between dynamic user interests and results within a universe of entities or ideas [15-17] apply an RNN where each neuron represents a dimension. The RNN with Deep Learning algorithm emulates the human brain within a management decision structure where generated judgements are taken in a structured way as a discriminative process based on a hierarchical process [18-19].

1.2 Research structure

Section 2 of this paper presents the generative adversarial network research background, GANs have been applied in different applications that require random samples from a common source of truth. In addition, there are several algorithms and mathematical models that generate a GAN. Section 3 defines the General Adversarial Random Neural Network mathematical model: Discriminator, Generator and learning algorithm. The Generative Adversarial RNN is validated for several neural configurations and vector dimensions for the Generator and Discriminator respectively. Section 4 presents the experimental results of two Discriminators: single output neuron and the same number of output neurons as the Generator. Conclusions and future work are shared in Section 5. The learning objective of the RNN Generator meets the RNN Discriminator expectations whereas the RNN Discriminator learning target identifies unfit individuals.

2 Research Background

Generative Adversarial Networks (GANs) estimate Generative models via an adversarial process in which two models are simultaneously trained. A Generative model "G" captures the data distribution and a Discriminative model "D" estimates the probability that a sample is from training data rather than G [1]. The training method for G is to maximize the probability of D making a mistake. New architectural features and training procedures improve the semi-supervised learning performance and improved sample generation of the GANs [20]; several techniques based on a heuristic understanding of the non-convergence problem are introduced to optimize the convergence of the GAN. An adversarial autoencoder is based on the variational inference between the aggregated of the hidden code vector of the autoencoder with an arbitrary prior distribution [21]; the adversarial autoencoder learns a deep generative model that maps the imposed prior to the data distribution.

Conditional adversarial networks are proposed as a general purpose solution to the image to image translation problems [22]; the networks learn the mapping from the input image to output image and a loss function to train this mapping. Stacked GANs generate high-resolution images [23]. The first GAN sketches the original shape and colours of a scene based on a given text description producing low-resolution images, the second GAN generates high-resolution images with photo-realistic details based on the outputs of the first GAN. GANs improve noise robustness of automatic speech recognition systems [24] when operating in GANs on log-Mel filterbank spectra instead of waveforms. GANs can effectively suppress additive noise in raw waveform speech signals, improving perceptual quality metrics.

An evolutionary GANs framework offers stable GAN training and improved generative performance [25]. The model evolves a population of generators to play the adversarial game with the discriminator where different adversarial training objectives are applied as mutation operations and each individual generator. GANs are also applied as unsupervised learning in clustering tasks. The cluster structure is not retained in the GAN latent space using traditional methods [26]. By sampling latent variables from a mixture of encoded variables and continuous latent variables, coupled with an inverse network trained jointly with a clustering specific loss achieve clustering in the latent space. Triangle Generative Adversarial Networks GAN consists of two generators and two discriminators for cross-domain joint distribution matching [27]. The generators learn the two-way conditional distributions between two domains, while the discriminators distinguish real data pairs and two kinds of fake data pairs. GAN mode collapse is the production of replicas with reduced diversity. To solve this issue, a discriminator is modified to make decisions based on multiple samples from the same class, either real or artificially generated [28]. A generative model for time-series data preserves temporal dynamics where new sequences include the original relationships between variables across time. A framework for generating realistic time series combines the flexibility of the unsupervised paradigm with the control afforded by supervised training [29].

3 The Generative Adversarial Random Neural Network

The Random Neural Network (RNN) [12-14] is a spiking recurrent stochastic model for neural networks that represents more closely how signals are transmitted in many biological neural networks where they travel as spikes or impulses, rather than as analogue signal levels (Fig. 1).

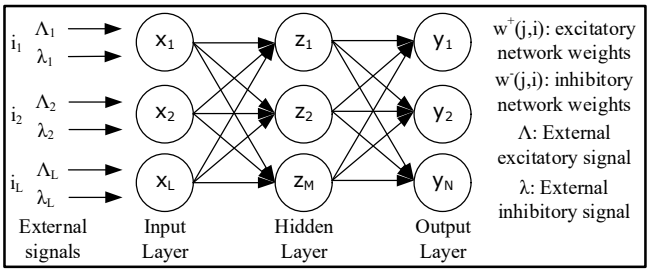

Fig. 1. The Random Neural Network structure

3.1 The Random Neural Network

The RNN is composed of P neurons where each neuron receives excitatory (positive) and inhibitory (negative) spike signals from external sources which may be L sensory impulses or P internal neurons:

- $X_L = (x_1, x_2, \dots, x_L)$, a variable L-dimensional vector $X \in [0,1]^L$ represents the input state q_L for the neuron L; where scalar L values range $1<L<\infty$;
- $Z_M = (z_1, z_2, \dots, z_M)$, a M-dimensional vector $Z \in [0,1]^M$ that represents the hidden neuron state q_M for the neuron M; where scalar M values range $1<M<\infty$;
- $Y_N = (y_1, y_2, \dots, y_N)$, an N-dimensional vector $Y \in [0,1]^N$ that represents the neuron output state q_N for the neuron N; where scalar N values range $1<N<\infty$.

These spike signals occur following independent Poisson processes of rates $\lambda^+(m)$ for the excitatory spike signal and $\lambda^-(m)$ for the inhibitory spike signal respectively, to neuron $m \in \{1,...P\}$ (Fig. 2).

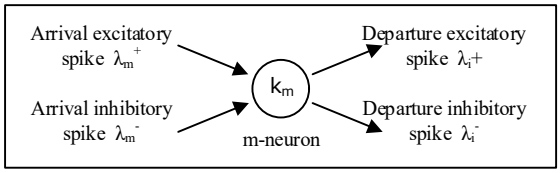

Fig. 2. The Random Neural Network

3.1.1 Principles

The state of the m neuron network at time t is represented by the vector of non-negative integers $k(t) = [k_1(t), \ldots k_m(t)]$ where $k_m(t)$ is the potential of neuron m at time t. Neurons interact with each other by interchanging signals in the form of spikes of unit amplitude:

- A positive spike is interpreted as an excitation signal because it increases by one unit the potential of the receiving neuron m, $k_m(t^+) = k_m(t) + 1$;
- A negative spike is interpreted as an inhibition signal decreasing by one unit the potential of the receiving neuron m, $k_m(t^+) = k_m(t) - 1$, or does not affect if the potential is already zero, $k_m(t) = 0$.

Each neuron i accumulates signals and it will fire if its potential is positive. Firing will occur at random and spikes will be sent out at rate r(i) with independent, identically and exponentially distributed inter-spike intervals:

- Positive spikes will go out to neuron j with probability $p^+(i,j)$ as excitatory signals;
- Negative spikes with probability $p^-(i,j)$ as inhibitory signals.

3.1.2 Model

Neuron i may send spikes out of the network with probability d(i). We have:

$$d(i)+\sum_{j=1}^{n}[p^+(i,j)+p^-(i,j)]=1 \quad \text{for } 1 \le i \le n \tag{1}$$

Neuron potential decreases by one unit when the neuron fires either an excitatory spike or an inhibitory spike (Fig. 3). External (or exogenous) excitatory or inhibitory signals to neuron i will arrive at rates $\Lambda(i)$, $\lambda(i)$ respectively by stationary Poisson processes. The RNN weight parameters $w^+(j,i)$ and $w^-(j,i)$ are the non-negative rate of excitatory and inhibitory spike emission respectively from neuron i to neuron j:

$$\begin{aligned} w^+(j,i) &= r(i)p^+(i,j) \ge 0 \\ w^-(j,i) &= r(i)p^-(i,j) \ge 0 \end{aligned} \tag{2}$$

Information in this model is transmitted by the rate or frequency at which spikes travel. Each neuron i, if it is excited, behaves as a frequency modulator emitting spikes at rate $w(i,j) = w^+(i,j) + w^-(i,j)$ to neuron j. Spikes will be emitted at exponentially distributed random intervals. Each neuron acts as a non-linear frequency demodulator transforming the incoming excitatory and inhibitory spikes into potential.

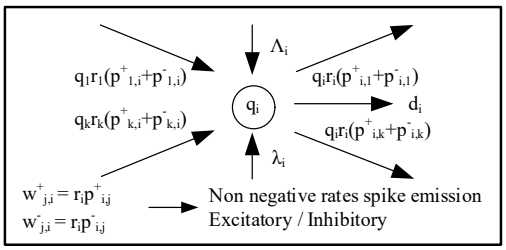

Fig. 3. The Random Neural Network model

This network model has a product form solution; the network's stationary probability distribution can be represented as the product of the marginal probabilities of the state of each neuron where q_i is the probability neuron i is excited.

3.1.3 Theorem

The probability distribution of the network state is defined as $p(k,t) = Prob[k(t) = k)]$ and the marginal probability a neuron i is excited at time t as $q_i(t) = Prob[k_i(t)>0]$. The stationary probability distribution $p(k) = \lim_{t \to \infty} p(k,t)$ and $q_i = \lim_{t \to \infty} q_i(t)$ where k(t) is a continuous time Markov chain that satisfies Chapman-Kolmogorov equations [12-14].

$$q_i = \frac{\lambda^+(i)}{r(i)+\lambda^-(i)} \tag{3}$$

$$r(i) = \sum_{j=1}^{n} [w^+(i,j) + w^-(i,j)] \quad \text{for } 1 \le i \le n \tag{4}$$

where the $\lambda^+(i)$, $\lambda^-(i)$ for i=1,…,n satisfy the system of nonlinear equations:

$$\lambda^+(i) = \sum_{j=1}^{n} \left[q_j r(j) p^+(j,i) \right] + \Lambda(i) \tag{5}$$

$$\lambda^-(i) = \sum_{j=1}^{n} \left[q_j r(j) p^-(j,i) \right] + \lambda(i) \tag{6}$$

3.1.4 Learning Algorithm

The RNN learning algorithm is based on the gradient descent of a quadratic error function. The backpropagation model requires the solution of n linear and n nonlinear equations each time the n neuron network learns a new input and output pair. Gradient Descent learning algorithm optimizes the network weight parameters W in order to learn a set of k input-output pairs (i,y) where successive inputs are denoted i =

$\{i_1,\ldots,i_k\}$ and the successive desired outputs are represented $y = \{y_1,\ldots,y_k\}$. Each input vector $i_k = (\Lambda_k, \lambda_k)$ is the pair of the excitatory and inhibitory signals entering each neuron: $\Lambda_k = [\Lambda_k(1),\ldots, \Lambda_k(n)]$, $\lambda_k = [\lambda_k(1),\ldots, \lambda_k(n)]$. Each output vector $y_k = (y_{1k},\ldots, y_{nk})$, $y_{ik} \in [0,1]$ is composed of the desired values of each neuron. The desired output vectors are approximated by minimizing the cost function E_k:

$$E_k=\frac{1}{2}\sum_{i=1}^{n} a_i\left(q_i-y_{ik}\right)^2 \qquad a_i \geq 0 \tag{7}$$

Each of the n neurons of the network is considered an output neuron. The function of the variable a_i is to remove neurons from the network output. The network learns both n x n weight matrices $W_k^+ = \{w_k^+(i,j)\}$ and $W_k^- = \{w_k^-(i,j)\}$ by calculating new values of the network parameters for each input $i_k = (\Lambda_k, \lambda_k)$.

3.2 The Generative Adversarial Random Neural Network

The Generative Adversarial Random Neural Network (RNN) is composed of two Random Neural Networks (Fig. 4) with different learning algorithms where l^k defines the latent space and d^k represents the true distribution respectively.

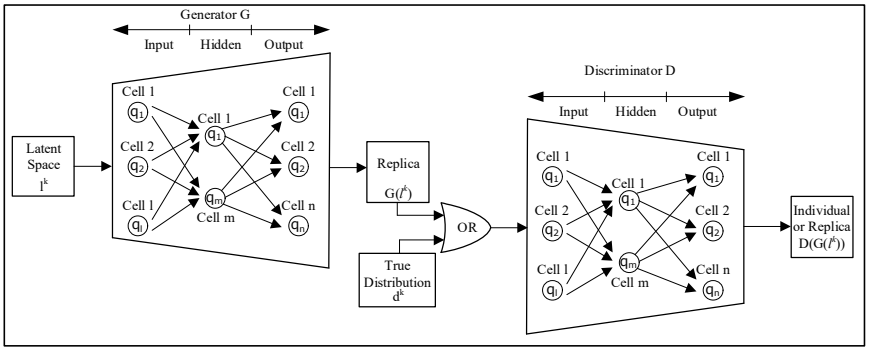

Fig. 4. Generative Adversarial Random Neural Network

3.2.1 Discriminator

The Discriminator D is defined as an RNN that consists of L input neurons, M hidden neurons and N output neurons as:

- $X_{DIS} = (x_{Dis\text{-}1}, x_{Dis\text{-}2}, \ldots, x_{Dis\text{-}L})$, a variable L-dimensional vector $X \in [0,1]^L$ represents the input state q_L for the neuron L; where scalar L values range $1<L<\infty$;
- $Z_{DIS} = (z_{Dis\text{-}1}, z_{Dis\text{-}2}, \ldots, z_{Dis\text{-}M})$, a M-dimensional vector $Z \in [0,1]^M$ that represents the hidden neuron state q_M for the neuron M; where scalar M values range $1<M<\infty$;

- Y_{DIS} = (y_{Dis-1}, y_{Dis-2}, … , y_{Dis-N}), a N-dimensional vector Y Є $[0,1]^N$ that represents the neuron output state q_N for the neuron N; where scalar N values range 1<N< ∞. The output neuron potential codify the probability that input is from true data distribution rather than the latent space;
- W_{DIS}(j,i) is the (L+M+N)x(L+M+N) matrix of network weights that represents the connection from neuron j to neuron i; where i Є [x_L, z_M, y_N] and j Є [x_L, z_M, y_N].

The learning objective of the Discriminator is to maximize the probability of assigning the correct label to both real individuals and replicas from the Generator, therefore, is trained to minimize the error of the real data distribution. The learning equation of the Discriminator and the rule for weight update takes the generic form:

$$W_{DIS}^{k}(i,j)=W_{DIS}^{k-1}(i,j) - \eta \sum_{i=1}^{n} \left(d_i^k - y_{DIS-i}^k\right) \left[\frac{\partial q_{DIS-i}}{\partial W_{DIS}(i,j)}\right]_k \quad (8)$$

where η is the learning rate, the term W_{DIS}(i,j) denotes either w^+(i,j) or w^-(i,j), d_i^k is the true distribution and q_{DIS-i} represents the output state of the discriminator neuron i respectively.

3.2.2 Generator

The Generator G is defined as an RNN that consists of L input neurons, M hidden neurons and N output neurons as:

- X_{GEN} = (x_{Gen-1}, x_{Gen-2}, … , x_{Gen-L}), a variable L-dimensional vector X Є $[0,1]^L$ represents the input state q_L for the neuron L; where scalar L values range 1<L< ∞. The input neuron potential represents the latent space or noise;
- Z_{GEN} = (z_{Gen-1}, z_{Gen-2}, … , z_{Gen-M}), a M-dimensional vector Z Є $[0,1]^M$ that represents the hidden neuron state q_M for the neuron M; where scalar M values range 1<M< ∞;
- Y_{GEN} = (y_{Gen-1}, y_{Gen-2}, … , y_{Gen-N}), an N-dimensional vector Y Є $[0,1]^N$ that represents the neuron output state q_N for the neuron N; where scalar N values range 1<N< ∞. The output of the generator represents the individual replica;
- W_{GEN}(j,i) is the (L+M+N)x(L+M+N) matrix of network weights that represents the connection from neuron j to neuron i; where i Є [i_L, z_M, y_N] and j Є [i_L, z_M, y_N].

The learning objective of the Generator is to minimize the error of the Discriminator, therefore, is trained based on the Discriminator output or probability of the true data. The number of output neurons of the Generator y_{Gen-N} is the same as the number of input neurons in the Discriminator x_{Dis-L}. The learning equation of the Generator and the rule for weight update take the generic form:

$$W_{GEN}^{k}(i,j)=W_{GEN}^{k-1}(i,j) - \eta \sum_{i=1}^{n} \left(y_{DIS-i}^k\right) \left[\frac{\partial q_{GEN-i}}{\partial W_{GEN}(i,j)}\right]_k \quad (9)$$

where η is the learning rate, the term $W_{GEN}(i,j)$ denotes either $w^{+}(i,j)$ or $w^{-}(i,j)$ and $q_{GEN\text{-}i}$ represents the output state of the generator neuron i respectively.

3.2.3 Learning Algorithm

The proposed learning algorithm is based on data from the latent space, l^k, and data from the true distribution, d^k (Fig. 4):

1) the Discriminator D learns $W_{DIS}^{k}(i,j)$ where $Y_{DIS}=D(d^k)$ represents the probability d^k is an individual data from the true distribution;
2) the Generator G creates a replica from the latent space $G(l^k)$;
3) the Discriminator D assesses the replica $Y_{DIS}=D(G(l^k))$;
4) the Generator G learns $W_{GEN}^{k}(i,j)$ based on Y_{DIS};
5) the method iterates on step 2) until either the Generator can not further learn to reduce the Discriminator probability of detection or the Discriminator can not distinguish between the replica and the real individual.

4 Experimental results

This section evaluates the Generative Adversarial RNN against two neural configurations:

- Generator: Learning rate of 0.1 with input and output vectors of five, ten and twenty dimensions;
- Discriminator: Learning rate of 1.0 with output vector of one and same as the Generator dimensions.

Experimental results are obtained based on 1,000 independent experiments with random inputs to the Generator and Discriminator rather than a prescribed dataset. The time series consists of 10,000 data points per experiment. Other GAN algorithms prosed in the literature review evaluate GANs as a classification method or image generator rather than as an autoencoder as proposed in this paper therefore there is not a direct comparison between algorithms.

Statistical information includes Standard Deviation and 95% Confidence Range. The variables represented cover the average error of the Discriminator after learning the individual with the required number of iterations and time. In addition, the average error of the Generator creating the replica is also shown with its validation time and the detection of the Discriminator. This detection corresponds to the output neurons of the Discriminator $y_{Dis\text{-}N}$.

4.1 Discriminator output neurons: One

The Discriminator learns at a very reduced error after very few iterations and detects replicas from the Generator with only one neuron on its output layer. The larger the number of generator neurons, the lesser learning iterations required; however its

learning time also increases. A Generator with larger neurons has a greater error in the replica and requires a longer learning time, this is due to the replica becomes more complex too. The error that the Discriminator feedbacks to the Generator is very reduced and almost constant to the Generator size (Table 1).

Table 1. Generative Adversarial RNN Validation – Discriminator: 1 neuron

Stage	Variable	Generator Neurons	Value	σ	95% CR
Discriminator	Error	5	4.78E-31	2.61E-31	1.62E-32
		10	4.12E-31	2.92E-31	1.81E-32
		20	3.34E-31	2.90E-31	1.80E-32
	Iteration	5	37.39	3.82	0.24
		10	28.01	3.87	0.24
		20	23.49	3.96	0.25
	Time (ns)	5	4.77E+06	2.45E+07	1.52E+06
		10	4.69E+06	1.53E+07	9.46E+05
		20	9.54E+06	1.82E+07	1.13E+06
Generator	Error Discriminator	5	1.64E-09	9.39E-07	4.48E-08
		10	5.62E-08	1.49E-05	7.49E-07
		20	5.07E-08	8.96E-05	5.26E-06
	Error Generator	5	1.09E-02	1.30E-02	6.21E-04
		10	1.83E-02	7.30E-03	3.66E-04
		20	2.18E-02	4.08E-02	2.39E-03
	Time (ns)	5	5.63E+09	1.18E+11	5.64E+09
		10	9.87E+09	1.25E+11	6.25E+09
		20	1.51E+10	4.91E+09	2.88E+08

The Discriminator with only one output neuron does not provide enough resolution for the Generator to learn suffering diminished gradient (Fig. 5). The Discriminator error converges to a constant value, independent of the Generator neurons.

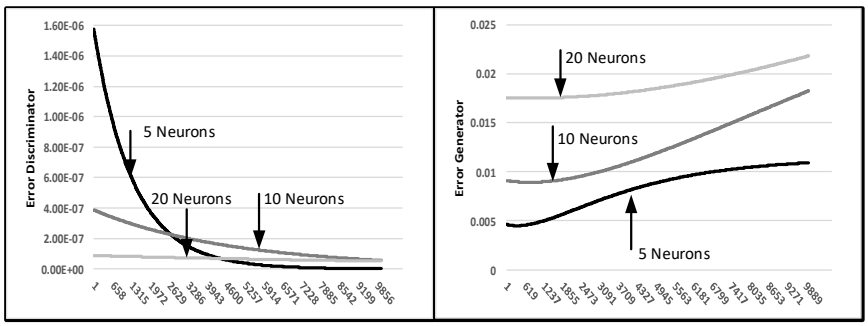

Fig. 5. Discriminator and Generator Replica error – Discriminator: 1 neuron

4.2 Discriminator output neurons: Same as Generator

The Discriminator with the same neurons as the Generator on its output layer detects replicas from the Generator although it requires more learning iterations than the pre-

vious validation due to the increase of the output neurons (Table 2). The number of learning iterations of the Discriminator is nearly constant against the Generator neurons, however, its learning time increases with the number of Generator neurons. The error of the Discriminator is still lower than the Generator, therefore outperforming its detection capabilities.

Table 2. Generative Adversarial RNN Validation – Discriminator: Generator neurons

Stage	Variable	Generator Neurons	Value	σ	95% CR
Discriminator	Error	5	8.29E-31	1.62E-31	1.01E-32
		10	8.38E-31	1.61E-31	9.97E-33
		20	8.22E-31	1.70E-31	1.06E-32
	Iteration	5	124.41	3.68	0.23
		10	124.22	2.71	0.17
		20	124.37	1.84	0.11
	Time (ns)	5	2.12E+07	3.59E+07	2.23E+06
		10	4.76E+07	5.62E+07	3.48E+06
		20	1.59E+08	5.79E+07	3.59E+06
Generator	Error Discriminator	5	3.29E-05	3.39E-05	2.10E-06
		10	3.33E-05	1.11E-05	6.90E-07
		20	3.27E-05	8.21E-06	4.65E-07
	Error Generator	5	1.67E-03	2.04E-03	1.26E-04
		10	3.40E-03	2.50E-03	1.55E-04
		20	6.68E-03	3.57E-03	2.02E-04
	Time (ns)	5	1.10E+09	4.82E+08	2.99E+07
		10	3.64E+09	2.15E+09	1.33E+08
		20	1.68E+10	6.13E+09	3.47E+08

The Discriminator provides a greater resolution to the Generator enabling its stable learning to converge to a non-optimum value suffering overfitting (Fig. 6). The error of the Discriminator reduces following the Generator learning where the more complex (larger dimensions) the Generator is, the greater the error.

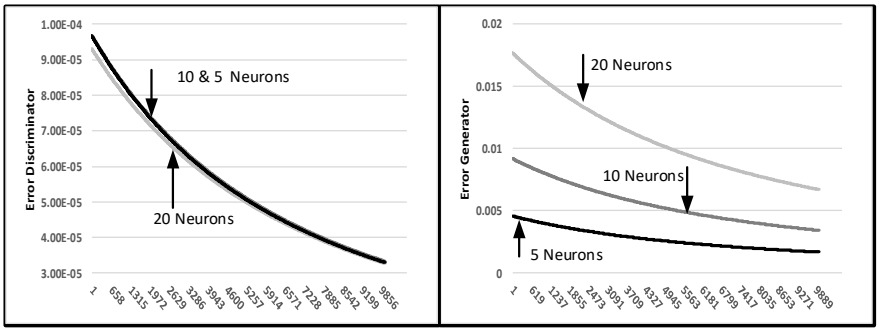

Fig. 6. Generative Adversarial RNN Validation – Discriminator: 5 neurons

5 Conclusions

This paper has presented the Generative Adversarial Random Neural Network (RNN) with the same features and use cases as a Generative Adversarial Network: an RNN Generator produces individuals mapped from a latent space while the GAN Discriminator evaluates them based on the true data distribution. The validation results are successful although not optimum: the learning objective of the RNN Generator creates replicas at low error whereas the RNN Discriminator learning target identifies unfit individuals. The number of output neurons in the Discriminator has a clear effect on the Generator learning capabilities.

The error that the discriminator feedbacks to the Generator (diminished gradient) may not be large enough for the Generator to learn (overfitting), although a large error may cause the Generator not to converge (instability). In addition, experimental results show that the larger the dimensions of the Generator, the worse the replica becomes, although this is due to its greater complexity.

Future work will include additional experiments to increase the dimensions of the input vector. The modifications within the Generator cost function obtained from the Discriminator feedback to enhancement its learning capabilities will be analysed. Finally, the addition of Deep Learning clusters will be assessed in terms of learning performance and quality for both the Generator and the Discriminator.

References

1. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems. 27, (2014), 1-9.
2. X. Wang, A. Gupta. Generative Image Modeling Using Style and Structure Adversarial Networks. European Conference on Computer Vision. (2018), 318-335.
3. L. Wang, S. Mostafavi, Y-S Ho, K-J Yoon. Event-Based High Dynamic Range Image and Very High Frame Rate Video Generation Using Conditional Generative Adversarial Networks. Proceedings of the IEEE / Computer Vision Foundation. Conference on Computer Vision and Pattern Recognition. (2019), 10081-10090.
4. A. Fadaeddini, B. Majidi, M. Eshghi. A Case Study of Generative Adversarial Networks for Procedural Synthesis of Original Textures in Video Games. International Digital Games Research Conference: Trends, Technologies, and Applications. (2018), 118-122.
5. N. Matsumura, H. Tokura, Y. Kuroda, Y. Ito, K. Nakano. Tile Art Image Generation Using Conditional Generative Adversarial Networks. International Symposium on Computing and Networking Workshops. (2018), 209-215.
6. K. Ak, J. Lim, J. Tham, A. Kassim. Attribute Manipulation Generative Adversarial Networks for Fashion Images. Proceedings of the IEEE Computer Vision Foundation. Conference on Computer Vision. (2019), 10541-10550.
7. M-C. Lee, B. Gao, R. Zhang. Rare Query Expansion Through Generative Adversarial Networks in Search Advertising. ACM International Conference on Special Interest Group on Knowledge Discovery and Data Mining. (2018), 500–508.
8. H. Emami, M. Dong, Nejad, C. Glide. Generating synthetic CTs from magnetic resonance images using generative adversarial networks. 45, 8, (2018), 3627-3636.

9. H. Husain, R. Nock, R. Williamson. A Primal-Dual link between GANs and Autoencoders. Advances in Neural Information Processing Systems. 32, (2019), 1-10.
10. P. Dallaire, N. Killoran. Quantum generative adversarial networks. Physics Review A 98, 012324. (2018), 1-8.
11. S. Lloyd, C. Weedbrook. Quantum Generative Adversarial Learning. Phys. Rev. Lett. 121, 040502, (2018), 1-5.
12. E. Gelenbe. Random Neural Networks with Negative and Positive Signals and Product Form Solution. Neural Computation, 1, (1989), 502-510.
13. E. Gelenbe: Stability of the Random Neural Network Model. Neural Computation, 2, 2, (1990), 239-247.
14. E. Gelenbe. Learning in the Recurrent Random Neural Network. Neural Computation, 5, (1993), 154-164.
15. W. Serrano, E. Gelenbe. Intelligent search with deep learning clusters. Intelligent Systems Conference (2017), 632-637.
16. W. Serrano. Neural Networks in Big Data and Web Search. Data, 4, 7, (2019), 1-41.
17. W. Serrano, E. Gelenbe, Y. Yin. The Random Neural Network with Deep Learning Clusters in Smart Search. Neurocomputing, 396, (2020), 394-405.
18. W. Serrano. Fintech Model: The Random Neural Network with Genetic Algorithm. Procedia Computer Science, 126, (2018), 537-546.
19. W. Serrano. Genetic and deep learning clusters based on neural networks for management decision structures. Neural Computing and Applications, 32, (2020), 4187–4211.
20. T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen. Improved techniques for training GANs. International Conference on Neural Information Processing Systems. (2016), 2234-2242.
21. A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey. Adversarial Autoencoders. International Conference on Learning Representations. (2016), 1-16.
22. P. Isola, J. Zhu, T. Zhou, A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. IEEE Conference on Computer Vision and Pattern Recognition. (2017), 5967-5976.
23. H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, D. Metaxas. StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 41,8, (2019), 1947-1962.
24. C. Donahue, B. Li, R. Prabhavalkar. Exploring Speech Enhancement with Generative Adversarial Networks for Robust Speech Recognition. IEEE International Conference on Acoustics, Speech and Signal Processing. (2018), 5024-5028.
25. C. Wang, C. Xu, X. Yao, D. Tao. Evolutionary Generative Adversarial Networks. IEEE Transactions on Evolutionary Computation. 23, 6, (2019), 921-934.
26. S. Mukherjee, H. Asnani, E. Lin, S. Kannan. ClusterGAN: Latent Space Clustering in Generative Adversarial Networks. Association for the Advancement of Artificial Intelligence. (2019), 4610- 4617.
27. Z. Gan, L. Chen, W. Wang, Y. Pu, Y. Zhang, H. Liu, C. Li, L. Carin. Triangle Generative Adversarial Networks. Advances in Neural Information Processing Systems. 30, (2017), 1-10.
28. Z. Lin, A. Khetan, G. Fanti, S. Oh. PacGAN: The power of two samples in generative adversarial networks. IEEE Journal on Selected Areas in Information Theory. 1, 1, (2020), 324-335.
29. J. Yoon, D. Jarrett, M. Schaar. Time-series Generative Adversarial Networks. Conference on Neural Information Processing Systems. (2019), 1-11.