Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 180 (2021) 676–685

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Industry 4.0 and Smart Manufacturing

# Taxonomy of generative adversarial networks for digital immunity of Industry 4.0 systems

Vagan Terziyan [a], Svitlana Gryshko [b], Mariia Golovianko [c]

[a]*Faculty of Information Technology, University of Jyväskylä, Jyväskylä, 40100, Finland*
[b]*Department of Economic Cybernetics, Kharkiv National University of Radioelectronics, Kharkiv, 61166, Ukraine*
[c]*Department of Artificial Intelligence, Kharkiv National University of Radioelectronics, Kharkiv, 61166, Ukraine*

**Abstract**

Industry 4.0 systems are extensively using artificial intelligence (AI) to enable smartness, automation and flexibility within variety of processes. Due to the importance of the systems, they are potential targets for attackers trying to take control over the critical processes. Attackers use various vulnerabilities of such systems including specific vulnerabilities of AI components. It is important to make sure that inappropriate adversarial content will not break the security walls and will not harm the decision logic of critical systems. We believe that the corresponding security toolset must be organized as a trainable self-protection mechanism similar to immunity. We found certain similarities between digital vs. biological immunity and we study the possibilities of Generative Adversarial Networks (GANs) to provide the basis for the digital immunity training. We suggest the taxonomy of GANs (including new architectures) suitable to simulate various aspects of the immunity for Industry 4.0 applications.



*Keywords:* Industry 4.0; Generative Adversarial Networks; cybersecurity; artificial digital immunity

## 1. Introduction and related work

Generative Adversarial Network (GAN) is a popular research concept and efficient (adversarial) machine learning (ML) paradigm. According to this concept, two neural networks are trained towards contradictory objectives (both useful for various applications), compete with each other, co-evolve while training and mutually facilitate each other`s training process. Since [1], where GANs are used for generating samples in an unsupervised way, a large number of GAN modifications and corresponding algorithms have been proposed. Gui et al. [2]


10.1016/j.procs.2021.01.290

recently published a review on various GANs methods from the perspectives of algorithms, theory, and applications. They also combined GANs with other ML algorithms (semi-supervised learning, reinforcement learning, and transfer learning) to be used in popular applications (processing image, speech, music and audio for generic data science and medicine in particular). The most promising artificial intelligence solutions and architectures, including GANs, are being successfully applied in Industry 4.0, which indicates the emerging trend of the digital (data-driven) transformation of manufacturing enterprises. Such applications working for industrial diagnostics, fault detection and predictive maintenance require essential amount of data about faults, which is not always available but may be generated using GANs [3]. GANs present a way to learn deep representations without extensively annotated training data. As shown in [4], this feature can be used in several applications (image style transfer, synthesis, semantic editing, super-resolution and classification). GANs can be also useful in the context of digital twins [5] or simulation-based decision support tools used for optimization and resilience of future factories [6]. Integrated with proper simulation environments [7, 8, 9, 10], where challenging (complex and adversarial) conditions can be modelled and digitally applied against various industrial assets and processes, the GAN-driven digital twins may essentially improve security and productivity of industrial processes. Good review of GANs with particular applications presented in [11]. For example, image synthesis could be handled by DCGAN [12]; engineering design could be addressed by GANs enhanced by augmented human intelligence and creativity [13]; for the imbalanced data classification, a class expert GAN (CE-GAN) can be useful [14]; various data quality issues could be handled by ACGAN as an auxiliary classifier GAN [15]; surface inspection could be provided using, e.g., LSGAN architecture, i.e., modified least-square GANs [16]; condition monitoring could be addressed by GANs with the denoising autoencoder as a discriminator as suggested in [17]; anomaly detection by f-AnoGAN architecture [18]; security of cyber-physical systems by novel CGAN (conditional GAN) architecture GAN-Sec [19], etc.

Domains related to security, remain the natural application area where GANs could be applied with the major impact. [20] published an interesting review on the application of GANs in cybersecurity. They observe two major approaches within cybersecurity studies: (a) use of GANs to improve generalization to unforeseen adversarial attacks, by generating novel adversarial samples as training data for ML models (i.e., improving the “defence” mechanism); (b) training GANs aiming to generate realistically looking adversarial data that can fool a security system (i.e., improving the “attack” mechanism). Most of the currently discussed cybersecurity strategies follow the paradigm of the proactive security, i.e., attempting to prevent potential attacks before they occur. As noticed in [21], many real-world applications still follow a reactive security approach (i.e., systems are updated when novel attacks are observed). The GAN concept enables proactive security solutions up to the extent of the digital immunity. Usually term “immunity” is applied in biology to define the balanced state of an organism, which is capable to apply adequate defences to fight unwanted biological invasion. Termanini [23] contributed to applying the immunization legacy to the digital world. He described the Cognitive Early Warning Predictive System as a digital immune system, which inoculates critical systems with the vaccination as a service and considers so-called “smart vaccine” (autonomous AI robot) as a holy grail of digital immunity. The early warning predictive component in the system is the AI reasoning engine that acquires knowledge from past attacks and predicts, probable incoming cyberattacks. Such approach to digital immunity applies top-down (expert-driven) AI (ontological representation of knowledge and inference engine). In this paper, we suggest an opposite, bottom-up (computational intelligence) AI approach to digital immunity and smart vaccination, which can be obtained as a result of adversarial GAN-driven learning.

Our generic objective or the meta-goal (for the former, present and future research) is further empowerment of the GAN models as an instrument capable to address the academic challenges related to strong AI, digital cloning, collective intelligence, etc., and also real world, practical and emerging challenges of Industry 4.0. The specific objective addressed in this paper is to suggest a set of GAN modifications suitable for the use as a proactive security (digital immunity) enabler for intelligent components within Industry 4.0.

The rest of the paper is organized as follows: In Section 2, we discuss on the analogy of GANs as abstract models with the immunity and vaccination as proactive security mechanism; in Section 3, we present the major modification for traditional GAN architecture by introducing the “Reality” component as the third one neural-net-driven player; in Section 4, we describe reasonable updates regarding traditional GAN players (“Discriminator” and “Generator”); in Section 5, we used the new properties of all the three GAN players to get the structural representation (taxonomy) of the potential new GAN architectures suitable to address the proactive security challenges; and we conclude in Section 6.

## 2. Digital immunity and generative adversarial networks

The brief story, on how we came to the immunity concept as a motivation to update further the GAN framework, is as follows. Our team (mainly AI experts) was interested in GAN as such because it has been one of the major inventions in AI and ML for the last decade. We worked out some techniques on using GANs to design digital cognitive clones of humans capable to replace the latter ones in some industrial processes [24]. Later, when we meet the challenges of NATO SPS project "Cyber-Defence for Intelligent Systems" (http://recode.bg/natog5511) and particularly security of AI/ML-supervised supply-chain and logistics, we discover that one of the most dangerous vulnerabilities of such systems (allowing the most insidious attacks) is the intelligent component and ML in particular [25]. Successful adversarial attacks to ML may cause huge impact as shown in [26], and, as indicated in [27] industry practitioners are still not equipped to protect, detect and respond to such attacks. ML systems are vulnerable both at the learning phase of the model design (e.g., data poisoning attack) or when the trained model start making predictions (evasion attacks). A common way to force the models to make wrong predictions is to design adversarial examples as discovered in [28] and such design involves perturbations that are almost indistinguishable to humans, yet force ML models to make wrong predictions. An adversarial example is an input to ML model that is intentionally designed by an attacker to fool the model into producing an incorrect output. Effective defenses as response to the new types of attacks appear slower than the new attacks are being invented. There is just one example among many similar ones: the popular and effective ML-based malware detection techniques (e.g., the ones based on Random Forest, Support Vector Machine, and Convolutional Neural Networks) has been shown to completely fail under the novel adversarial texture malware perturbation attack presented in [29]. Therefore, a reactive security paradigm cannot be exploited alone without adding also some proactive security efforts to be prepared for potential attacks before they occur, by incorporating knowledge of the attacker's strategy into the defense mechanism [21]. We believe that good analogy for the proactive security in real world would be an immune system of a human and its interaction with the vaccines as proactive security measures. Such interaction can be modelled with GANs, which is a good instrument to represent and pre-train two coevolving counterparts (artificial attacker and artificial defender). Our interests on GANs development and the need for proactive security solution for our projects meet at that point, and we discovered several updates to the traditional GAN architecture, which might be suitable to design some sort of digital immunity for the intelligent systems.

The core of the GAN-driven generic abstract skeleton for the digital immune system is the basic filtering element, which we call a "Debunker", which corresponds to the Discriminator (D) in GAN model. The task of it is to address an input (e.g., incoming "stranger") from the external environment and to make a decision about the origin and the intentions of the input (either true / real input ("welcome guest") from the environment or a falsified input ("intruder") from an attacker pretending to be a real one). To train the "immunity" in such adversarial situations, D must have some supervised interaction experience with the two external players: (a) "Reality" component, which represents the real external environment and selects true ("healthy") inputs (data samples) from it. In some cases, we need the Reality to generate inputs as challenging as possible for further processing by other components; and (b) Fake component, which corresponds to the Generator (G) in GAN architecture, acts as an attacker and generates intruders or false inputs (artificially generated data samples) aiming to fool D, break into the system and make harm there. It is assumed that D does not know the origin of the inputs (Reality or Fake) during training process and must decide itself, whether it is a real or a fake input. The training is possible due to the mechanism of feedbacks for all the players (D, Reality and G). This mechanism enables players to constantly improve own performance in achieving their competing goals. Therefore, after the training, D will be trained to protect (aka immunity), G – to attack (aka vaccine) and Reality – to be as challenging as possible for both rivals.

Consider the classical formulation of the GAN problem. D gets samples from two sources: (a) from Reality, assuming existence of $\{x^{(1)}, \dots, x^{(m)}\}$ minibatch of reality samples with probability distribution $p_{data}(x)$; (b) from G, assuming existence of $\{z^{(1)}, \dots, z^{(m)}\}$ minibatch of fake samples with probability distribution $p_z(z)$. Discriminator is trained to distinguish between the fake and the reality samples. Generator tries to generate samples from scratch aiming at the same distribution as the reality samples distribution. It learns to capture the reality samples distribution and therefore to fool the Discriminator. The fundamental mathematical model of GANs in the classical formulation reproduces the minimax game between a generator (G) and a discriminator (D) with the objective function $V(D, G)$.

In game theory terms, the GAN model converges when D and G reach a Nash equilibrium, which is the optimal point for the minimax equation below:

$$min_G max_D V(D,G) = E_{x\sim p_{data}(x)}[log\ D(x)] + E_{z\sim p_z(z)}[log\ (1 - D(G(z))]. \quad (1)$$

The equation above includes two addends that reflect the competing goals of the players: (a) the entropy of the real data samples passing through D (i.e., the probability of the "successful" job of D when it allows the flow of data from the reality ($p_{data}(x)$); (b) the entropy of the random data samples fed to D, i.e. the "failure" level of D when it allows the flow of fake data created by G ($p_z(z)$). Naturally, D works to maximize the function (1) trying to increase the first term (i.e., successfully recognize the real data inputs) and also increase the second one (i.e., uncover the fake inputs from G). G has an opposite task. Its cheating is aimed at ensuring that D is making mistakes: misses the "correct" samples (minimizing the first addend) and allows the flow of fakes (minimizing also the second term). Therefore, G works to minimize the function (1). Loss function for D (provided as a feedback for update of D) considers the own misclassification error and, therefore, the generation success of G:

$$Loss(D) = \frac{1}{m} \cdot \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]. \quad (2)$$

Loss function for G (provided as a feedback for update of G) takes into account the own generation error (uncovered samples) and, therefore, the discrimination success of D:

$$Loss(G) = \frac{1}{m} \cdot \sum_{i=1}^{m} log(1 - D(G(z^{(i)}))). \quad (3)$$

Training GANs means training simultaneously two competing neural networks using the loss values as a feedback for the backpropagation [30]. Training algorithm for D starts with fixing (temporally "freezing") the weights of G and taking samples from both real world and generated images. Then D is being trained to distinguish between the R and G images. Then the time is given for G to learn. At the first stage, training algorithm for G freezes the weights of D. Then samples from G are created. The backpropagation error through D is used to update the weights of G. There can be several iterations of such asynchronous (D-G) training. At each iteration, the gradient of the loss function is updated, as well as the parameters to min (max) the loss function (gradient descent / ascent). These two steps iterate until convergence. G is "perfect" if it is impossible to discriminate between real and generated input samples (D accuracy ≈ 0.5). In this paper, we show that the basic principles of GANs are well suited for creation of different digital "immunity" and "vaccination" models for the AI-driven Industry 4.0 systems.

## 3. Reality as the third player in GAN architecture

Unlike in the classical GAN-architecture with two major players (G vs. D), a third full-fledged independent player representing reality is needed to model the immunity. In such architectures, the "Reality" would be an independent, proactive and learnable player, as the two others. The Reality component is responsible for supplying the real-world data as a training content for G and D. We distinguish three main ways of organizing such a delivery:

- Method (A) "Reality as Such", when an unsupervised flow of data from the natural environment is fed to the system (similar to traditional GAN architecture);
- Method (B) "Filtered Reality", when a controlled flow of data from the natural environment is fed to the system. To do this, a new player "Reality Recruiter" is introduced within the GAN architecture. Its task is to select the appropriate data set (correct from the point of view of the "immune response" creation) from the environmental data samples (from either the most representative or the "trickiest" ones);
- Method (C) "Generated Reality", when the generated data flow from the simulated or artificial environment is fed to the system. "Reality as Such" (A) version, in which the "Reality" component is not an intelligent one (neither learnable, nor a neural network driven). This component serves as the "Reality" model according to the "Closed World Assumption". Most ML algorithms process training sets as "closed world" systems. Following

the "Closed World Assumption", one concludes that this information is false due to the lack of information about the sample. That means the unknown simply cannot be true by default. An alternative to it is the Open World Assumption, according to which lack of information does not imply the missing information to be false.

Assume we have a complete set of the Real-World samples (e.g. images), which have some distribution $\theta = p_{data}(x)$. The "Reality as Such" component randomly selects samples from the Real World with respect to the distribution $\theta$. This component is the same as the one used in the classical GAN and it is not an active player in the "game": Fake (G) vs. Debunker (D), i.e., it does not depend on the feedback from this game.

"Filtered Reality" (B) version of the "Reality" component is an intelligent one (capable of learning and neural network driven). This component intends to represent the "Reality" so that it would be difficult to learn how to fake it. It also follows the Closed World Assumption (see Figure 1).

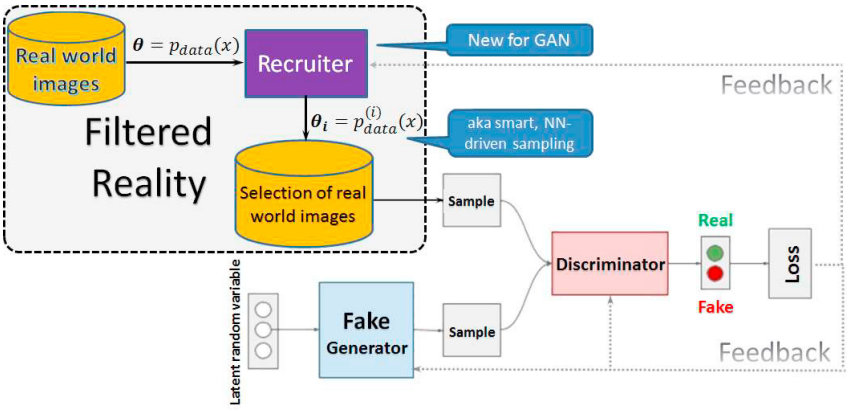


Fig. 1. GAN with Filtered Reality (enhanced with the trainable "Recruiter").

Assume we have a complete set $R$ of the Real-World samples (e.g. images), which has some distribution $\theta = p_{data}(x)$. Such set can be divided into a number of subsets $R_1\ R_2, \dots, R_n \in \mathcal{P}(R)$ so that each subset $R_i$ has some distribution $\theta_i = p^{(i)}_{data}(x)$. The core of Filtered Reality is a neural network "Reality Recruiter" (RR). Its configuration works as a filter, which recognizes a sample of a particular distribution $\theta_i$ and "recruits" it as a set $R_i$ to represent the Reality. After the set $R_i$ is collected, the traditional "game": G vs. D starts. It lasts as long as necessary for reaching the balance (i.e., G learns $\theta_i$ and is capable to fool D). Then RR gets the feedback from this game as a loss function, which is the inverse ratio to the time needed to reach the balance. After one cycle of the backpropagation, RR will be modified and will be capable to recruit the Reality samples with different distribution $\theta_j$, which will be harder to fake. After several iterations RR is expected to find such configuration $\theta_{target}$, which corresponds to the subset of samples $R_{target}$ from the reality, which would be most difficult to forge. Loss function for RR has the following basic structure:

$$Loss(RR) = \frac{1}{n_{balance}}, \tag{4}$$

where $n_{balance}$ is the amount of training (G vs. D) iterations needed to reach the balance.

RR must function so that the distribution of the samples in the subset from the real world images created/updated by it would be as difficult as possible to forge, and, in case of such an attempt, it would be as easy as possible to recognize a counterfeit. D in such architecture tries to distinguish between real and fake images according to the discovered difference between corresponding distributions. G tries to guess the distribution of the current subset of the real-world images and generates fake images of the world according to that distribution. RR takes and constantly updates the subset of the real-world images and, therefore, updates the distribution of images (aiming to hide this distribution from G as long as possible). The Feedback loop for RR works as follows. First, G learns how to fool D (i.e., the distribution of generated images is almost the same as the distribution of the real images subset) forcing it to guess (fifty-fifty). Then R will get a feedback as a loss function proportional to G and D overall learning feedback during the last iteration. RR will update its weights so that the distribution of the new subset of the real-world

images increases the "hacking" (G) effort for the next iteration and decreases the "discriminating" (D) effort. RR forms (and later during training) constantly updates the selection criteria, which is an explicit specification of the features for the distribution $\theta_i$. On the basis of the selection of the real-world images (with the distribution $\theta_i$) the traditional GAN (with the two players: G vs. D) starts to work until the balance of G vs. D performances is reached. Assume that the balance is reached in $n_i$ iterations. RR, which is a neural network, receives the feedback from GAN (after the balance is reached). The feedback is a loss function $Loss(RR)$ value, which depends on how easy the previous selection made by RR can be hacked by G. Therefore, following the basic formulation (4):

$$Loss(RR) = \frac{1}{n_i}. \tag{5}$$

The loss function initiates the backpropagation process in the RR network and, as the result, the network reconfigures the parameters of the distribution from $\theta_i$ to $\theta_{i+1}$ . Then RR prepares a new group of training samples as a subset of all the real-world images according to the updated distribution $\theta_{i+1}$. The specifics of the neural network training guarantees that this new selection will be more difficult to hack ($n_{i+1} \geq n_i$) for G than the previous one. Therefore iteratively (after a number of iterations) RR will find an optimal (most protected against potential evasion attacks) subset of training data and also both D and G will be trained in the most challenging environment.

The third possible player for the Reality role is the "Generated Reality", which is also the learnable and neural network driven. This component intends to generate some new version of the "Reality" so that it would be as difficult as possible to learn how to fake. It follows the Open World Assumption, capable of working without the set of the Real-World samples (e.g., images). It has a new component Reality Generator (GR), which generates (at each iteration) a set of samples as the representation of the generated reality with some distribution $\theta_i = p_z^{(i)}(z)$. Then the traditional "game": Fake (G) vs. Debunker (D) starts and lasts as long for reaching the balance as necessary (i.e., G learns $\theta_i$ and is capable to fool D with fakes). Then the neural network of the GR gets the feedback from this game as the loss function value, which is the aggregated difference between D (ally) losses and G (adversary) losses during the time needed to reach the balance. After one step of the backpropagation, the GR network will be modified and will be capable to generate the Reality samples with a different distribution $\theta_j$, which is harder to fake. With the Open World Assumption this iterative process may continue infinitely or can be stopped on some satisfactory configuration $\theta_{target}$, which will correspond to the subset of the generated (difficult enough to forge) samples from the reality. The Reality Generator ($GR$) loss function is:

$$Loss(GR) = \sum_{i=1}^{n_{balance}} [Loss_i(D) - Loss_i(G)], \tag{6}$$

where $i$ indicates the iteration number and $n_{balance}$ is the amount of training (G vs. D) iterations needed to reach the balance. To avoid the dependence of the $n_{balance}$ in formulas (4) and (6) and $n_i$ in formula (5) on the random weight initialization within D and potential mode collapse issues [31], GAN must be trained synchronously with several initializations for D and the obtained values $n_{balance}^{(1)}, n_{balance}^{(2)}, \dots, n_{balance}^{(k)}$ for the discrete time (measured in iterations) needed to reach the balance can be aggregated using the harmonic mean function [32], which is less tolerant to the high valued outliers.

GR must function so that the distribution created/updated by it would be as difficult as possible to forge, and, in the case of the attempt, it would be as easy as possible to recognize a counterfeit. Tasks of D are the same as in the previous GAN architectures. D tries to distinguish between real and fake images according to the discovered difference between corresponding distributions. G tries to guess the distribution of the real-world images and generates fake images of the world according to that distribution. GR makes changes ("creatively") in the real world and, therefore, updates the distribution of real-world images (aiming to hide this distribution from G as long as possible). The feedback loop for the GR works as follows. First, G finishes learning how to fool D (i.e., the distribution of generated images is almost the same as the distribution of real images) forcing it to guess (fifty-fifty). Then the GR gets feedback as a loss function proportional to the G's D's overall learning feedback during the last iteration. Finally, the GR updates its weights so that new distribution of the real-world images increases the "hacking" (G) effort for the next iteration and decreases the "discriminating" (D) effort. We name these adversarial learning techniques the Immunity Training. Each of three GAN components perform its tasks regarding the immunity development:

- RR (also GR) creates a variety of sophisticated "vaccines" against potential attacks on the "organism";
- D simulates the immune system of the organism, which helps the organism to perform its functionality correctly after the sophisticated attacks;
- G helps the vaccine to attack the vulnerabilities of the organism in the most challenging way to activate the immune system towards positive evolution.

## 4. Updating classical GAN players for immunity development

The immune system is created in a repeated confrontation between a defender (D) and an attacker (G). G is a sparring partner for D and its role is to simulate the attacks to train D and strengthen the digital immune system as a whole. We distinguish two principal ways to model possible attacks with the help of the G, and, accordingly, two types of the "Fake"/G player (Figure 2 (left)):

1) "Generated Fake", which generates fake samples;
2) "Generated Noise", when the samples are real, but they are subjects to the poisoning by some noise.

Fake-component of the first type is defined as "Generated Fake". This version of the "Fake" component is exactly G from the traditional GAN architecture. This is an intelligent (learning and neural network driven) component. G tries to generate samples from scratch (from the random noise $z$) with the distribution $p_z(z)$ aiming to bring this distribution as close as possible to the reality samples distribution. It trains to capture the reality samples distribution and, therefore, to fool D. Therefore, its target is to minimize its own loss function (3).

Fake-component of the second type is defined as "Generated Noise" (GN). Such component is not creating fakes as such, but it creates (generates) special signals ("noise"), which being mixed with the Reality samples, keep these samples to be naturally looking for humans and at the same time confusing for the artificial D (evasion attack). We consider two kinds of "noise": the one, which is added as such to the reality samples; or the one, which is served as a "style", i.e., set of features values, which are integrated to the deeper representation of the reality sample.

Similarly, to the traditional G component, the GN trains to capture the reality samples distribution and therefore to fool any kind of D. Therefore, its target is to minimize own loss function:

$$Loss(GN) = \frac{1}{m} \cdot \sum_{i=1}^{m} log\left(1 - D\left(GN\left(z^{(i)}\right)\right)\right). \quad (7)$$

Finally, let us consider the element, which ensures security functions of the digital immunity, i.e., the Debunker (D in terms of GAN). We distinguish two principal types of security modelling (Figure 2 (right)):

I) "Traditional Discriminator" based on AI;
II) "Turing Discriminator" based on collective intelligence.

The Traditional Discriminator (I) version of the "Debunker" component is exactly the one, named D in the traditional GAN architecture. This is an intelligent (learning and neural network driven) component. D tries to distinguish between samples coming from the "Reality" channel from the ones coming from the "Fake" channel, i.e., it learns the decision boundary between real and fake samples. When trained, it can be used as a component, which "protects" the "organism" of an intelligent system from the dangerous alien instances. While training its target is to minimize its own loss function (2).

The second type of the Debunker is an interesting, innovative and potentially very useful and multi-purpose component. The "Turing Discriminator" (TD) version of the "Debunker" component behaves, on the one hand, as a "black box", i.e., similar to the traditional D, it has the same structure of the loss function, however, on the other hand, it has different semantics. TD is a collective intelligence, which includes at least one human (H), and at least one traditional learning neural (D) discriminator. Both (H and D) are differentiating between input samples ("real" or "fake"). The TD outputs the probability distribution between "Match" and "No match" of the H and D opinions, which is used as the loss function to train D. The D player in TD tries to learn to synchronize its own opinions on the inputs with the H's opinions. Such schema allows training D to be resistant against potential evasion attacks. Sometimes it is possible to use a "strong previously trained artificial classifier" AI-H instead of H, and then D tries to learn to the level of AI-H.

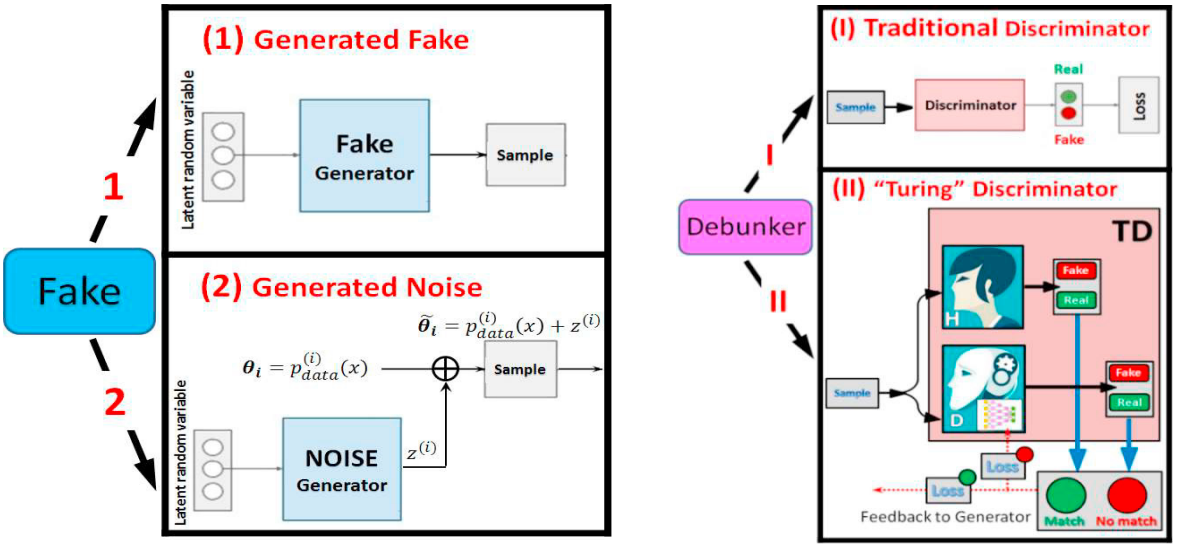


Fig. 2. Players (1, 2) for the "Fake" role (left). Players (I, II) for the "Debunker" role (right).

## 5. Periodic table of elements for GAN

The versatility of each of the GAN components creates unique opportunities for the adaptation of the digital immunity concept to the features of each particular system and its tasks. To order all possible combinations of elements, let us use the principles of the chemical "Periodic table of elements" (Table 1). Let`s briefly describe the features and denote the area of use for some of these GAN-based architectures of digital immunity.

Code (A) (1) (I) means that the AI immune system consists of Reality as Such, Generated Fake and Traditional Discriminator. (A) (1) (I) is a traditional GAN, which is a good illustration of a very basic generic "immunity" training model. G generates well-camouflaged and constantly evolving aliens for the "data poisoning" attacks, and D improves own performance in the discovery (a "friend" or an "alien") for every intruder.

Table 1. "Periodic table of elements": coding GAN-architectures for digital immunity.

| DEBUNKER | FAKE | |
|---|---|---|
| | "Generated Fake" (1) | "Generated Noise" (2) |
| REALITY: "Reality as Such" (A) | | |
| "Traditional Discriminator" (I) | (A) (1) (I) | (A) (2) (I) |
| "Turing Discriminator" (II) | (A) (1) (II) | (A) (2) (II) |
| REALITY: "Filtered Reality" (B) | | |
| "Traditional Discriminator" (I) | (B) (1) (I) | (B) (2) (I) |
| "Turing Discriminator" (II) | (B) (1) (II) | (B) (2) (II) |
| REALITY: "Generated Reality" (C) | | |
| "Traditional Discriminator" (I) | (C) (1) (I) | (C) (2) (I) |
| "Turing Discriminator" (II) | (C) (1) (II) | (C) (2) (II) |

GAN architecture with TD encoded as (A) (1) (II) is an illustration of a "vaccination" aiming protection of the intelligent systems against evasion attacks. TD improves own performance in being at least as good as a human in handling perturbated images. This architecture can be used also to train "cognitive clones" of particular human skills or other intelligent decision-makers.

GAN architecture encoded as (A) (2) (I) is a good illustration of a training model (aka "immunity against poisons") for an Intelligent System. GN generates well camouflaged and constantly evolving noise (or style) to compromise data, and D improves own performance in the discovery ("poisoned" or "clean") for every input.

Architecture encoded as (A) (2) (II) naturally inherits the properties of both (A) (2) (I) and (A) (1) (II) architectures. Architecture encoded as (B) (1) (I) is similar to (A) (1) (I) architecture but with the better trained attacker and, as a result, the better trained immunity. G generates constantly evolving aliens (imitating different representations of the reality) for the poisoning attacks; therefore, the generic immunity driver D (after training) will have more advanced capability in the discovery (a "friend" or an "alien") for every intruder. Architecture (B) (1) (II) naturally inherits the properties of both (A) (1) (II) and (B) (1) (I) architectures. Architecture (B) (2) (I) naturally inherits the properties of (A) (2) (I) and (B) (1) (I) architectures. Architecture (B) (2) (II) naturally inherits the properties of both (A) (2) (II) and (B) (2) (I) architectures. (C) (1) (I) is similar to (A) (1) (I) architecture but with the better trained attacker and, as a result, the better trained immunity. G generates constantly evolving aliens for the poisoning attacks (trying to capture not only the distribution of the reality samples but also the hidden model of how the reality is being modified by the GR component). Therefore, the generic immunity driver D will have more advanced capability after such training. Architecture (C) (1) (II) naturally inherits the properties of (C) (1) (I), (A) (1) (II) and (B) (1) (II) architectures. Architecture (C) (2) (I) naturally inherits the properties of (C) (1) (I), (A) (2) (I) and (B) (2) (I) architectures. Architecture (C) (2) (II) naturally inherits the properties of (C) (2) (I), (A) (2) (II) and (B) (2) (II) architectures. This architecture assumes that the GN will learn how to generate a suitable poison for images of a self-creative smart reality aiming at a sophisticated evasion attacks on the test data of an intelligent system. TD in this case learns to detect sophisticated malicious inputs.

As we see, different combinations of the key elements create 12 basic architectures of the AI-immunity. They cover a wide range of aspects to protect AI-driven Industry 4.0 systems from a variety of the reality manipulation attacks with varying degrees of difficulty.

## 6. Conclusions

Digital immunity as a basic trained self-protection mechanism is needed for Industry 4.0 systems. GAN is a suitable paradigm to simulate training environment needed for the digital immunity. We suggest a couple of modifications to the basic GAN schema, i.e., Reality as a proactive trained neural-network-driven player and Turing Discriminator, which enables human-in-the-loop. By combining various players in the GAN configuration (three options for the Reality role; two options for the Discriminator role and two options for the Generator role), we present 12 GAN architectures. Each of these architectures has certain specifics if applied for the purpose of security of Industry 4.0 systems. Tasks related to the immunity training against various sophisticated attacks (data poisoning, evasion, etc.) could be addressed by thorough choice of the appropriate GAN architecture. Smart and evolving adversarial components in the architectures essentially facilitate the immunity training. Basic GAN components can be multiplied (e.g., several Discriminators or Generators, or several humans within Turing Discriminator), which creates wide possibilities to create complex GANs for new challenging problems. Therefore, as the immediate goals for the future research would be constructing and testing GAN architectures with multiple human and AI components, i.e., with the embedded collective intelligence and checking the potential impact of such digitalized collaborative intelligence for a variety of industrial problems.

## References

[1] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y., 2014. Generative Adversarial Nets. In: Z. Ghahramani et al., eds. Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, 2672–2680.

[2] Gui J., Sun Z., Wen Y., Tao D., Ye J., 2020. A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. arXiv:2001.06937

[3] Lee Y., Jo J., Hwang J., 2017. Application of Deep Neural Network and Generative Adversarial Network to Industrial Maintenance: A Case Study of Induction Motor Fault Detection. In: Proceedings of the IEEE International Conference on Big Data, pp. 3248-3253. IEEE. doi:10.1109/BigData.2017.8258307

[4] Creswell A., White T., Dumoulin V., Arulkumaran K., Sengupta B., Bharath A. A., 2018. Generative Adversarial Networks: An Overview. IEEE Signal Processing Magazine, 35(1), 53-65.

[5] Booyse W., Wilke D. N., Heyns S., 2020. Deep Digital Twins for Detection, Diagnostics and Prognostics. Mechanical Systems and Signal Processing, 140, 106612.

[6] Bécue A., Maia E., Feeken L., Borchers P., Praça I., 2020. A New Concept of Digital Twin Supporting Optimization and Resilience of Factories of the Future. Applied Sciences, 10(13), 4482.

[7] Longo F., 2012. Supply Chain Security: an Integrated Framework for Container Terminal Facilities. International Journal of Simulation and Process Modelling, 7(3), 159-167.

[8] Bruzzone A., Massei M., Longo F., Poggi S., Agresta M., Bartolucci C., Nicoletti L., 2014. Human Behavior Simulation for Complex Scenarios based on Intelligent Agents. Proceedings of the 2014 Annual Simulation Symposium, pp. 1-10. Tampa (USA).

[9] Petrillo A., Felice F.D., Longo F., Bruzzone A., 2017. Factors Affecting the Human Error: Representations of Mental Models for Emergency Management. International Journal of Simulation and Process Modelling, 12(3-4), 287-299.

[10] Padovano A., Longo F., Nicoletti L., Mirabelli G., 2018. A Digital Twin Based Service Oriented Application for a 4.0 Knowledge Navigation in the Smart Factory. IFAC-PapersOnLine, 51(11), 631-636.

[11] Kusiak A., 2020. Convolutional and Generative Adversarial Neural Networks in Manufacturing. International Journal of Production Research, 58(5), 1594-1604.

[12] Radford A., Metz L., Chintala S., 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv:1511.06434.

[13] Radhakrishnan R., Bharadwaj V., Manjunath V., Srinath R., 2018. Creative Intelligence – Automating Car Design Studio with Generative Adversarial Networks (GAN). In: Machine Learning and Knowledge Extraction. Lecture Notes in Computer Science, Cham: Springer, Vol. 11015, 160–175.

[14] Cenggoroa T. W., 2018. Deep Learning for Imbalance Data Classification Using Class Expert Generative Adversarial Network. Procedia Computer Science, 135, 60–67.

[15] Shao S., Wang P., Yan R., 2019. Generative Adversarial Networks for Data Augmentation in Machine Fault Diagnosis. Computers in Industry, 106, 85–93.

[16] Wang K., Zhang X., Hao Q., Wang Y., Shen Y., 2019. Application of Improved Least-Square Generative Adversarial Networks for Rail Crack Detection by AE Technique. Neurocomputing, 332, 236–248.

[17] Wang Z., Wang J., Wang Y., 2018. An Intelligent Diagnosis Scheme Based on Generative Adversarial Learning Deep Neural Networks and its Application to Planetary Gearbox Fault Pattern Recognition. Neurocomputing, 310, 213–222.

[18] Schlegl T., Seeböck P., Waldstein S. M., Langs G., Schmidt-Erfurth U., 2019. f-AnoGAN: Fast Unsupervised Anomaly Detection with Generative Adversarial Networks. Medical Image Analysis, 54, 30-44.

[19] Chhetri S.R., Lopez A.B., Wan J., Al Faruque M.A., 2019. GAN-Sec: Generative Adversarial Network Modeling for the Security Analysis of Cyber-Physical Production Systems. In: Proceedings of the Design, Automation and Test in Europe Conference, 770–775.

[20] Yinka-Banjo C., Ugot O. A., 2020. A Review of Generative Adversarial Networks and its Application in Cybersecurity. Artificial Intelligence Review, 53, 1721–1736.

[21] Biggio B., Fumera G., Roli F., 2014. Pattern Recognition Systems under Attack: Design Issues and Research Challenges. International Journal of Pattern Recognition and Artificial Intelligence, 28(7), 1460002.

[22] Termanini R., 2016. The Cognitive Early Warning Predictive System Using the Smart Vaccine: The New Digital Immunity Paradigm for Smart Cities and Critical Infrastructure. CRC Press.

[23] Bau D., Zhu J.Y., Wulff J., Peebles W., Strobelt H., Zhou B., Torralba, A., 2019. Seeing what a GAN Cannot Generate. In: Proceedings of the IEEE International Conference on Computer Vision, 4502-4511.

[24] Terziyan V., Gryshko S., Golovianko M., 2018. Patented Intelligence: Cloning Human Decision Models for Industry 4.0. Journal of Manufacturing Systems, 48 (Part C), 204-217.

[25] Terziyan V., Golovianko M., & Gryshko S., 2018. Industry 4.0 Intelligence under Attack: From Cognitive Hack to Data Poisoning. In: K. Dimitrov, ed. Cyber Defence in Industry 4.0 Systems and Related Logistics and IT Infrastructure. NATO Science for Peace and Security Series - D: Information and Communication Security, vol. 51, 110-125.

[26] Ren K., Zheng T., Qin Z., Liu X., 2020. Adversarial Attacks and Defenses in Deep Learning. Engineering, 6(3), 346-360.

[27] Kumar R.S.S., Nyström M., Lambert J., Marshall A., Goertzel M., Comissoneru A., Xia S., 2020. Adversarial Machine Learning--Industry Perspectives. arXiv preprint arXiv:2002.05646.

[28] Szegedy C., Zaremba W., Sutskever I., Bruna J., Erhan D., Goodfellow I., Fergus, R., 2013. Intriguing Properties of Neural Networks. arXiv preprint arXiv:1312.6199.

[29] Liu X., Zhang J., Lin Y., Li H., 2019. ATMPA: Attacking Machine Learning-based Malware Visualization Detection Methods via Adversarial Examples. In: Proceedings of the IEEE/ACM 27th International Symposium on Quality of Service, Phoenix, AZ, USA, 1-10.

[30] Salimans T., Goodfellow I., Zaremba W., Cheung V., Radford A., Chen X., 2016. Improved Techniques for Training GANs. In: Advances in Neural Information Processing Systems (pp. 2234-2242).

[31] Bau D., Zhu J.Y., Wulff J., Peebles W., Strobelt H., Zhou B., Torralba, A., 2019. Seeing what a GAN Cannot Generate. In: Proceedings of the IEEE International Conference on Computer Vision, 4502-4511.

[32] Ferger W.F., 1931. The Nature and Use of the Harmonic Mean. Journal of the American Statistical Association, 26(173), 36-40.