Artificial Intelligence Review
https://doi.org/10.1007/s10462-019-09717-4

A review of generative adversarial networks and its application in cybersecurity

Chika Yinka-Banjo[1] · Ogban-Asuquo Ugot[1]

Abstract

This paper reviews Generative Adversarial Networks (GANs) in detail by discussing the strength of the GAN when compared to other generative models, how GANs works and some of the notable problems with training, tuning and evaluating GANs. The paper also briefly reviews notable GAN architectures like the Deep Convolutional Generative Adversarial Network (DCGAN), and Wasserstein GAN, with the aim of showing how design specifications in these architectures help solve some of the problems with the basic GAN model. All this is done with a view of discussing the application of GANs in cybersecurity studies. Here, the paper reviews notable cybersecurity studies where the GAN plays a key role in the design of a security system or adversarial system. In general, from the review, one can observe two major approaches these cybersecurity studies follow. In the first approach, the GAN is used to improve generalization to unforeseen adversarial attacks, by generating novel samples that resembles adversarial data which can then serve as training data for other machine learning models. In the second approach, the GAN is trained on data that contains authorized features with the goal of generating realistic adversarial data that can thus fool a security system. These two approaches currently guide the scope of modern cybersecurity studies with generative adversarial networks.

Keywords Cybersecurity · Deep Learning · Generative adversarial networks

1 Introduction

Deep neural networks build upon the architecture of the multilayer perceptron by increasing the number of hidden layers between the input and output layer. With an improvement in modern optimization techniques and regularization techniques (Kingma and Ba 2014), deep neural networks are capable of learning highly complex and large data representations. Research in Deep neural networks have resulted in the design and development of supervised learning models such as the Convolutional Neural Network, and the Recurrent Neural Network. Unsupervised learning architectures include the Boltzmann machine, Self-Organizing

✉ Ogban-Asuquo Ugot
ogbanugot@gmail.com

[1] Department of Computer Science, University of Lagos Akoka, Akoka, Lagos, Nigeria

Published online: 01 June 2019

maps and Autoencoders. An important class of deep neural networks is the generative models (Goodfellow 2016). Given an original training sample, the goal of the generative model is to learn the density function that describes the original sample data and then use this estimated density to generate fake yet realistic looking data, similar to the original sample (Goodfellow 2016). In terms of architecture and design specifications, these generative models are very similar to the models mentioned earlier in the supervised and unsupervised class. For most of the generative models, it is an intuitive modification to the cost function used for training and the training algorithm that gives these models their generative abilities.

Generative Adversarial Networks (GANs) are a type of deep generative model that directly estimates a density function over a data distribution using alternative training techniques. The core idea of a GAN is to train two adversarial networks a generator and a discriminator, in a form of minimax game. The goal of the generator is to generate realistic images that can fool the discriminator, whereas, the discriminator tries to classify generated images (generated by the generator network) as fake and classify the real images from the original sample as real (Goodfellow et al. 2014a, b). This minimax game is an alternative and intuitive means of estimating the density function of the original sample images (Goodfellow 2016).

GANs have been used in state-of-the-art realistic generation tasks such as video frame prediction (Lotter et al. 2016), improving image resolution (Ledig et al. 2016), generative image manipulation (Zhu et al. 2016) and image to text translation (Isola et al. 2017). In all these applications and many others not mentioned, GANs have proven to be the state of the art for generating sharp and realistic images. In this paper, we review an important yet relatively novel application of GANs in the cybersecurity space. Ultimately, the review covers two important areas;

1. Studies where GANs have been used to strengthen a Cybersecurity system.
2. Studies where GANs have been used in adversarial attacks on a cybersecurity system.

Cybersecurity is a broad term, that usually encompasses the study and design of security protocols that protects digital systems connected over the internet. In this review we shall take note of studies that have a focus on the security of a system being protected from adversarial attacks using a GAN or being attacked using a GAN. The umbrella term "Cybersecurity" is used because in today's world, almost any digital system can be accessed over the internet and therefore will require cybersecurity policies.

Section 2 of this paper presents a detailed review of generative modeling and ends with a review of Generative adversarial networks. In Sect. 3, the review of the two approaches for the GAN is presented in terms of applications involving strengthening a security system or posing adversarial attacks to a security system. Sections 4 and 5 conclude the review.

2 Generative modelling

Generative modelling is an active area of deep learning research that centers around realistic data generation. In Goodfellow (2016), maximum likelihood is identified as a central task around which generative models can be compared. Not all generative models use maximum likelihood.

$$\theta^* = \arg\max \mathbb{E}_{x\sim p_{\text{data}}} log P_{\text{model}}(x/\theta). \tag{2.1}$$

In Eq. 2.1, x is a vector describing the input, $P_{\text{model}}(x/\theta)$ is a density function controlled by the parameter θ. Maximum likelihood consists of measuring the log probability that the

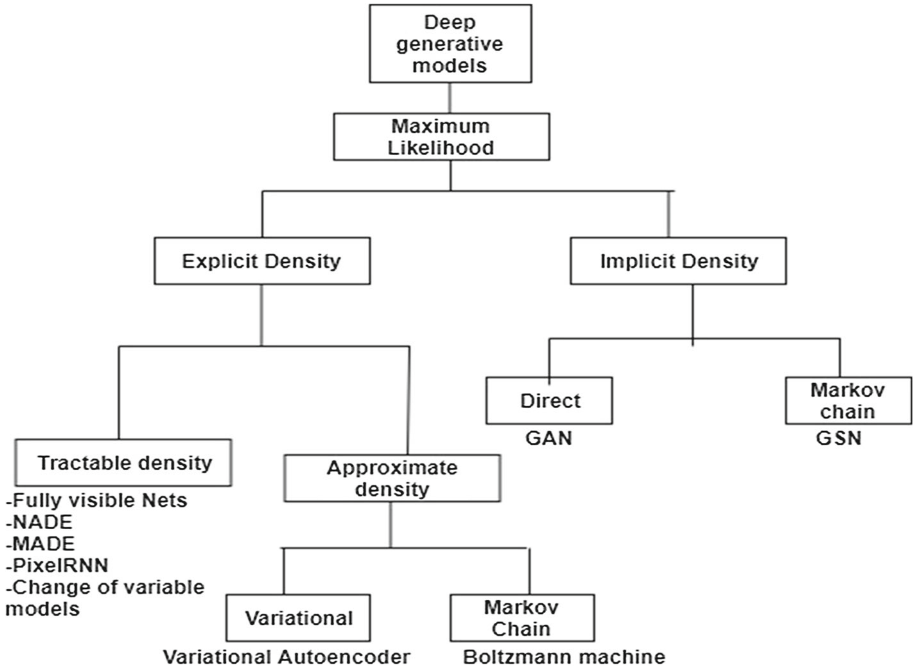


Fig. 1 Taxonomy of Generative models. Reproduced from Goodfellow (2016)

density function assigns to all data points and adjusting the parameter θ to increase that probability.

The way the generative models in Fig. 1 achieve maximum likelihood is what makes generative models different from each other (Goodfellow 2016). In the following subsections, the different branches of the generative model taxonomy with respect to maximum likelihood is reviewed.

2.1 Explicit density

We can start by grouping generative models into two primary classes, explicit and implicit density functions. Generative models that make use of an explicit density function actually define P_{model} and we are able to evaluate it and vary its parameters on the training data. Some density functions are intractable such as those over distributions like natural images and speech recognition and are therefore challenging to define. In such scenarios, an approximate density function is used. Based on this, we can further classify explicit density into tractable and approximate subsets.

2.1.1 Tractable density

2.1.1.1 Fully visible belief nets (FVBN) Introduced by Frey et al. (1996), fully visible belief nets make use of an explicit formula based on the chain rule of probability to decompose the probability distribution over a vector into a product over each of the members of the vector.

$$P_{\text{model}}(x) = P_{\text{model}}(x_1) \prod_{i=2}^{n} P_{\text{model}}(x_1 | x_i, \ldots, x_{i-1}) \tag{2.2}$$

In Eq. 2.2, we write the probability distribution of x_1 and then we multiply by the distribution over x_2 given x_1 and so on until we finally have a distribution over the final member of the vector. There have been substantial improvements and modern implementations to FVBNs since the original work by Frey et al. (1996), the most successful of these being the PixelCNN (Oord et al. 2016). Other recent implementations of FVBNs such as WaveNet (Van Den Oord et al. 2016), have been used for audio generation. As an advantage, FVBNs such as WaveNet have been proven to generate high quality data. One major disadvantage of FVBN is the fact that generating a sample is quite slow. This is usually within a time cost of $\mathrm{O}(n)$ for sample generation. Another important disadvantage is that generation is not controlled by a latent code. Many other generative models have a latent code that we can sample first describing the entire vector to be generated (Goodfellow 2016).

2.1.1.2 Change of variables The other major family of explicit tractable density models is the family of models based on the change of variables, an example is the Nonlinear ICA (Hyvärinen and Pajunen 1999). Using change of variables, we begin with a simple distribution like a gaussian and then use a nonlinear function to transform that distribution into another space.

$$y = g(x) \Rightarrow px(x) = py(g(x)) \left| \det\left(\frac{\partial g(x)}{\partial x} \right) \right| \tag{2.3}$$

In Eq. 2.3, using the nonlinear function $g(x)$ we transform a latent space containing x into a space of natural images.

The most significant drawback of these family of generative models is that one must carefully design the model to be invertible and must have a tractable determinant of the Jacobian. Therefore, the latent variables must have the same dimension as the data, this puts constraints on the flexibility of designing models.

2.1.2 Approximate density

According to Goodfellow (2016), this class of generative models are those that have intractable density functions but then use tractable approximations to these density functions.

2.1.2.1 Variational autoencoder (VAE) Variational Autoencoders (VAE) is one of the primary models belonging to the family of approximate density (Kingma and Ba 2014; Rezende et al. 2014). The basic idea is to write down a density function $\log p(x)$ as shown in Eq. 2.21.

$$\log p(x) \geq \log p(x) - D_{\mathrm{KL}}(q(z)||p(z|x)) = E_{z\sim q} \log p(x, z) + H(q) \tag{2.4}$$

From Eq. 2.4, the density function is intractable because we need to marginalize out the random variable z. The variable z is a vector of latent variables that provide a hidden code describing the input image. The challenge of intractability means we are forced to use a variational approximation that introduces a distribution q over the latent variable z to the extent that this distribution q is closer to the true posterior over the latent variable. The distribution q enables us to make a bound that becomes smaller and does a better job of establishing a lower bound that defines the true density. Variational encoders therefore define a lower bound;

$$L(x; \theta) \leq \log p\text{model}(x; \theta) \tag{2.5}$$

The most obvious challenge with this method is that the models tend to be asymptotically inconsistent unless q and thus the lower bound is near perfect (Kingma et al. 2016). Another important setback is the quality of generated data, were samples tend to have lower quality.

2.1.2.2 Markov chain approximations Another major family of models is the Boltzmann machine (Hinton and Sejnowski 1986). These models also make use of an intractable explicit function. In this case, the Boltzmann machine is defined by an energy function and the probability of a particular state is proportional to e raised to the power of that function. In order to convert this value to an actual probability, it is necessary to renormalize by dividing the sum over all the different states and that sum becomes intractable. Using Monte Carlo approximation methods (Goodfellow et al. 2016), we are able to approximate this density function.

There are challenges with using Monte Carlo approximations, one of them being the functions failing to mix between different modes. Also, Markov chain Monte Carlo methods perform poorly in high dimensional spaces because the Markov chains break down for very large images. Because of this, it is rare to see Boltzmann machines being used for image modelling task such as we've seen with GANs and other generative models so far.

2.2 Implicit density

Some models can be trained without even needing to explicitly define a density function. These models instead offer a way to train the model while interacting only indirectly with p_{model}, usually by sampling from it. These constitute the second branch, on the right side, of our taxonomy of generative models depicted in Fig. 1. Some of these implicit models based on drawing samples from p_{model} define a Markov chain transition operator that must be run several times to obtain a sample from the model. From this family, the primary example is the generative stochastic network (Bengio et al. 2014). As discussed in Markov chain Approximations, Markov chains often fail to scale to high dimensional spaces and impose increased computational costs for using the generative model. GANs were designed to avoid these problems. Finally, the rightmost leaf of our taxonomic tree is the family of implicit models that can generate a sample in a single step. At the time of their introduction, GANs were the only notable member of this family, but since then they have been joined by additional models based on kernelized moment matching (Li et al. 2015; Bengio et al. 2014).

2.3 Generative adversarial networks (GANs)

All the observations about generative models have set the stage for discussing GANs. There are four basic properties which set GANs aside from some of the other generative models. We outline these properties as follows (Goodfellow et al. 2016);

- Latent code: GANs make use of a latent code. The latent code describes everything that is generated later by the generator network. This property is common with VAEs and Boltzmann machines. The use of the latent code gives GANs an advantage over FVBNs.
- Asymptotically consistent: Unlike VAEs, GANs are asymptotically consistent. Although this is dependent in part on if we are able to find the equilibrium point of the game describing a GAN. According to Goodfellow (2016), asymptotical consistency is guaranteed once you've found the equilibrium point of the game and thus the true probability distribution that describes the data.

- No Markov chains: GANs do not require Markov chains neither to train the GAN nor to draw samples from the training data. We know that Markov chain (Monte Carlo) techniques set constraints on the performance of the restricted Boltzmann machine in high dimensional spaces. GANs have been proven to do well in high dimensional spaces (Radford et al. 2015) without relying on Markov chain Monte Carlo techniques.
- High quality generated data: GANs have often been regarded as producing the best samples, yet, there is no empirical way of quantifying this (Theis et al. 2015).

2.4 How GANs work

The basic framework of a GAN (Fig. 2) consists of two different neural networks that exists as adversaries of each other in the sense of game theory (Goodfellow et al. 2016). The operation of the two networks is defined by a game that has a well-defined payoff function. Each neural network tries to determine how well it can get the most payoff. Within the framework, the two different neural networks are known as the *Generator* and *Discriminator*. The proceeding subsections describe the generator and discriminator in more detail.

2.4.1 Generator

The generator G is the primary model we are interested in learning. It is the model that actively generates samples that are intended to resemble those that are in the original training distribution. The generator function is a differentiable function that has parameters that can

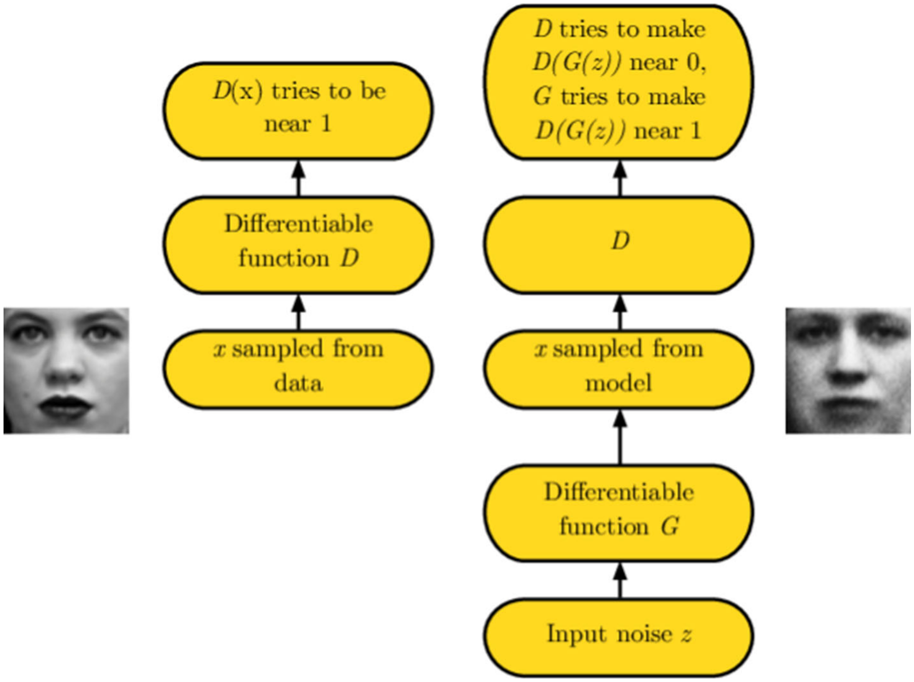

D(x) tries to be near 1
Differentiable function D
x sampled from data
D tries to make $D(G(z))$ near 0, G tries to make $D(G(z))$ near 1
D
x sampled from model
Differentiable function G
Input noise z

Fig. 2 The GAN framework. The GAN framework pits two adversaries against each other in a game. Each player is represented by a differentiable function controlled by a set of parameters. *Source*: Goodfellow (2016)

be learned using gradient decent. It is usually represented as a deep deconvolutional neural network (deconv net), but in principle, it can be any other model that has the differentiable property (Goodfellow 2016).

Symbolically, we can describe the generator network as follows;

$$x = G\left(z; \theta^{(G)}\right) \tag{2.6}$$

In Eq. 2.6, z is the vector that contains the latent code and x is the observed variable generated by the generator. We usually have every member of x depend on every member of z. Also, it is important to ensure that x has a higher dimension than z. According to (Goodfellow 2016), this ensures that the generator is not forced to generate a low dimensional manifold within the space of x.

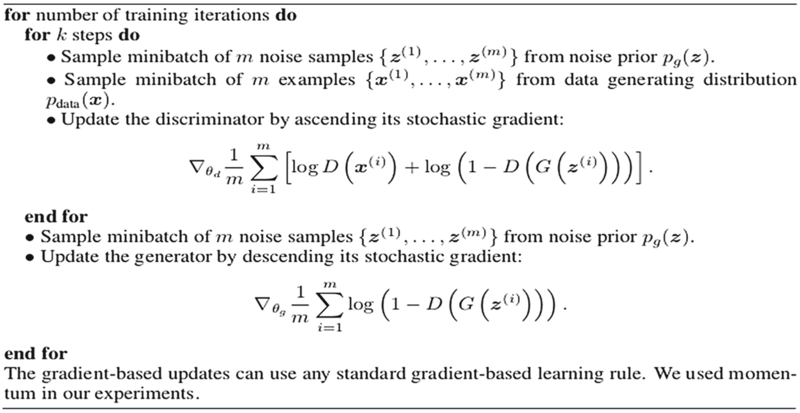
2.4.2 Discriminator

The discriminator D is not really necessary once the generator is successfully able to generate realistic data. The primary role of the discriminator is to inspect a sample to see whether the sample looks real or fake. Similar to the generator, the discriminator is any kind of differentiable function that has parameters we can learn using gradient decent. It is usually represented as a deep convolutional neural network (Zeiler and Fergus 2014). The discriminator's goal is to output a value near 1 for images from the original training set and values close to 0 for images from the generator.

2.5 Training a GAN

The training process shown in algorithm 2.1 and Fig. 3, consists of sampling a minibatch from both the training set and the generated samples, and then running the discriminator on those inputs. We begin by sampling the latent vector z from the prior distribution over the latent variables, z is essentially a vector of unstructured noise. It allows the generator to output a wide variety of different vectors (Goodfellow et al. 2016).

Algorithm 2.1: Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter (source: Goodfellow et al (2014)).



```
for number of training iterations do
  for $k$ steps do
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Sample minibatch of $m$ examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
    • Update the discriminator by ascending its stochastic gradient:
        $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right].$
  end for
  • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
  • Update the generator by descending its stochastic gradient:
      $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right).$
end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.
```

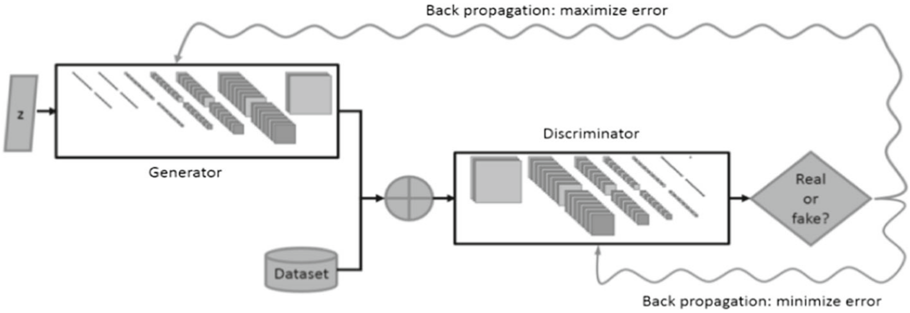

Back propagation: maximize error
z
Generator
Discriminator
Real or fake?
Dataset
Back propagation: minimize error

Fig. 3 The basic Generative Adversarial Network architecture

We then apply the generator function to the vector z. The generator outputs a sample that is then applied to the discriminator. The discriminator outputs a value which is essentially a binary classification of real or fake. The error loss on the discriminator's output is calculated using a cross entropy cost function. This error is then backpropagated to both the generator and the discriminator networks. Training stops when the discriminator can no longer discriminate between a generated data and training data. This point is known as the saddle point of the discriminator loss, and in principle should be the global minimum.

The nature of the cost function for both the discriminator and generator is discussed in more detail in Sect. 2.8. We use a Stochastic Gradient Decent (SGD) algorithm like ADAM on the two minibatches simultaneously and the gradient decent is carried out for both networks cost simultaneously.

2.6 Cost function for the GAN

One of the reasons GANs are quite straightforward to train is that we never actually try to infer the probability distribution $p(z/x)$, instead we sample values of z from the prior and then we sample values of x from $p(x/z)$ (Goodfellow et al. 2016). The other factor that appears to be crucial in the success of the training algorithm is the minimax game and the cost functions defined for the two neural networks. Each network has its own cost function as shown below in Eqs. 2.7 and 2.8;

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}} log D(x) - \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}} log(1 - D(G(x))) \tag{2.7}$$

$$J^{(G)} = -J^{(D)} \tag{2.8}$$

In Eq. 2.7, $J^{(D)}$ is the cost function for the discriminator and it is the cross entropy between the discriminator's prediction and the correct labels in the binary classification task discriminating between real and fake data. Another way to view $J^{(D)}$ is that it provides the cross entropy for predictions for both the real data from the training set and the generated samples from the generator. The generator cost $J^{(G)}$ in Eq. 2.8 simply minimizes the log probability of the discriminator cost function represented as the negated value of $J^{(D)}$. One can think of this as having a single value the generator is trying to minimize and the discriminator is trying to maximize (Goodfellow et al. 2016).

2.6.1 Non-saturating game

If we always have to negate the discriminator cost function as shown in Eq. 2.8, then whenever the generator is failing to fool the discriminator, the generator will have no gradient (or an extremely small value close to zero) because the output of the discriminator has saturated. Instead of negating $J^{(D)}$, Goodfellow (2016), proposes a solution to this problem by changing the order of the arguments to the cross-entropy function. This means that rather than trying to minimize the log probability of the right answer given by the discriminator, we have the generator trying to maximize the probability of the wrong answer.

At this point it is no longer possible to describe the equilibrium with just one function as shown in Eqs. 2.7 and 2.8. The generator cost function is now defined as follows;

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_z \log D(G(z)) \tag{2.9}$$

Both functions in Eq. 2.8 and 2.9 monotonically decrease in the same direction but are steep in different places. Goodfellow (2016) advocates for the use of the cost function in Eq. 2.9 for the generator network, because it solves the problem of non-saturating game (Radford et al. 2015).

2.7 Variant GAN architectures

This section discusses some notable GAN architectures with the most emphasis on the Deep Convolutional Generative Adversarial Network (DCGAN). GANs are scaled to large images using a hand designed process that makes use of Laplacian pyramids (Denton et al. 2015) to separate the image into multiple scales and generate each scale independently. More recently, the architecture introduced by Radford et al. (2015) known as Deep Convolutional Generative Adversarial Networks (DCGANs) have been used for generating large images with high dimensions.

The DCGAN places more emphasis on having multiple convolutional layers and using techniques such as batch normalization (Ioffe and Szegedy 2015) on every layer except for the last layer of the generator network, this makes learning more stable.

The overall network structure of the DCGAN is mostly borrowed from the all-convolutional net (Springenberg et al. 2015). This architecture contains neither pooling nor unpooling layers. When the generator needs to increase the spatial dimension of the representation in order to achieve better resolution, it uses the transposed convolution with a stride greater than 1. The basic idea of convolution is that one is sliding K filters, which can be thought of as K stencils, over the input image and produce K activations—each one representing a degree of match with a particular stencil. The inverse operation of convolution, known as the transposed convolution would be to take K activations and expand them into a preimage of the convolution operation. The intuitive explanation of the transposed convolution (Zeiler et al. 2010) is therefore, roughly, image reconstruction given the stencils (filters) and activations (the degree of the match for each stencil) and therefore at the basic intuitive level we want to expand each activation by the stencil's mask and add them up. The transposed convolution is also known as deconvolution (Zeiler et al. 2010).

The InfoGAN is another notable GAN architecture used for encoding meaningful features from the input data in the noise vector z in an unsupervised manner (Chen et al. 2016). InfoGANs are usually used in situations where the dataset is not very complex, where class labels are not available and where one wants to see what are the main meaningful features of a dataset and have control over them (Higgins et al. 2016).

The Wasserstein GAN (WGAN), is motivated to a large extent by the problem of convergence when training a GAN (Arjovsky et al. 2017). One problem with the GAN is that there is no direct correlation between the loss function and the quality of the generated data. The loss sometimes appears to be highly unstable and bares no reasonable indication of the true performance of the GAN. To solve this problem, the WGAN introduces the Wasserstein distance into the loss function for training and this results in a much more stable loss function that correlates with the quality of the generated data (Gulrajani et al. 2017).

2.8 Tuning a GAN

This section presents however briefly, some of the ways through which improvements to the GANs performance can be. It has been shown that labels may improve subjective sample quality because the ability to learn a conditional model $p(y/x)$ often gives much better samples from all classes than learning $p(x)$ does (Salimans et al. 2016). Also, batch normalization in G can cause strong intra-batch correlation. Finally, balancing the neural network architectures of the Discriminator and Generator may aid overall performance.

2.9 Summary of GANs

GANs are generative models that use supervised learning to approximate an intractable cost function, much as Boltzmann machines use Markov chains to approximate their cost and VAEs use the variational lower bound to approximate their cost. GANs can use this supervised ratio estimation technique to approximate many cost functions, including the KL divergence used for maximum likelihood estimation. GANs are relatively new and still require some research to reach their potential. In particular, training GANs requires finding Nash equilibria in high dimensional, continuous, non-convex games. Researchers should strive to develop better theoretical understanding and better training algorithms for this scenario. Success on this front would improve many other applications, besides GANs. GANs are crucial to many different state-of-the-art image generation and manipulation systems, and have the potential to enable many other applications in the future.

3 Cybersecurity and GANs

This section presents the literature concerning generative adversarial networks within the context of cybersecurity and adversarial attacks on system security. The review is presented in two sections, the first Sect. 3.1 presents a review of some studies where GANs have been used to abate adversarial attacks and strengthen a security system's performance against adversarial attacks. The second Sect. 3.2 presents a review of studies where GANs have been used to create adversarial systems and thus, to study the potential GANs have as adversarial threats to security systems.

3.1 Improving adversarial detection and security

Ever since the invention of GANs in 2014, focus on using the technology for adversarial studies has been one of the key research areas for the application of GANs. The motivation for this is due to the fact neural networks and other machine learning models in general perform poorly when there are changes in the feature space within which learning occurs (Malhotra 2018). The problem of generalization can be seen as a weakness that can used by adversarial

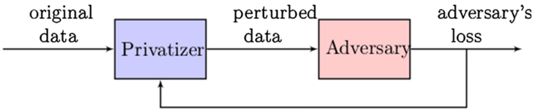


Fig. 4 Generative Adversarial Privacy. *Source*: Huang et al. (2018s)

systems to trick machine learning models into making wrong decisions (Goodfellow et al. 2014a, b).

The process of improving a system's security against adversarial attack begins with knowing what type of attacks to expect. In supervised classification tasks, adversarial attacks that aim to increase the likelihood of misclassification by injecting poison samples into a training data are the so called poisoning attacks. Evasion attacks on the other hand, consists of manipulations and distortions to the input data to evade accurate classification by a trained classifier (Szegedy et al. 2013; Biggio and Roli 2018).

One way through which researchers have tried to improve performance against poisoning and evasion adversarial attacks is by training machine learning models to be prepared for such attacks (Apruzzese et al. 2018; Lin et al. 2018). This is done by including examples of possible poison samples in the training dataset (Anderson et al. 2016). These adversarial examples are generated using generative adversarial networks from original training examples (Yin et al. 2018). The effect of this is a resulting increase in robustness against adversarial attacks, given that the machine learning model has learned some examples of potential adversarial attacks within a particular domain (Grosse et al. 2017). Significant increase in malware detection have been achieved using GAN generated adversarial examples in training sets (Kim et al. 2018) and in steganography for securely encoding data (Shi et al. 2017).

So far, we have presented literature were the research goal is generally centered around the generation of adversarial samples which can then be used as training data for another machine learning model, such as a classifier. There are also important security studies worth discussing were the generator network having learned an adversarial embedding space, is used for tasks such as stenography and cryptography (Abadi and Andersen 2016). Consider the work in (Tang et al. 2017), here, the researchers develop a GAN based stenographic scheme by simulating the rivalry between steganography with additive distortion and deep-learning based steganalysis. The Automatic Stenographic Distortion Learning (ASDL-GAN) scheme works by relying on the generator to produces steganographic images according to the learnt distortion function, while the steganalytic discriminator classifies between the cover and the produced steganographic images. In studies such as (Zügner et al. 2018), the steganographic distortion embedding space would serve as potential adversarial perturbation to otherwise normal image data. Instead, the authors in (Tang et al. 2017), train the generator to learn this distortion space and thus use it for steganographic work.

Finally, we conclude this section by briefly highlighting research on data privacy that is based on similar techniques discussed in the previous paragraph. The strength of data privacy and utility depends on the ability to incorporate noise where it matters. Researchers in Huang et al. (2018) present a technique termed Generative Adversarial Privacy (GAP), an innovative context-aware model of privacy that allows the GAN learn privatization mechanisms from the dataset without requiring access to the dataset statistics (Fig. 4).

3.2 Creating adversarial systems

On the other hand, important studies have been carried out by researchers looking at ways GANs can be used by adversarial systems to spoof or trick security systems. Research in

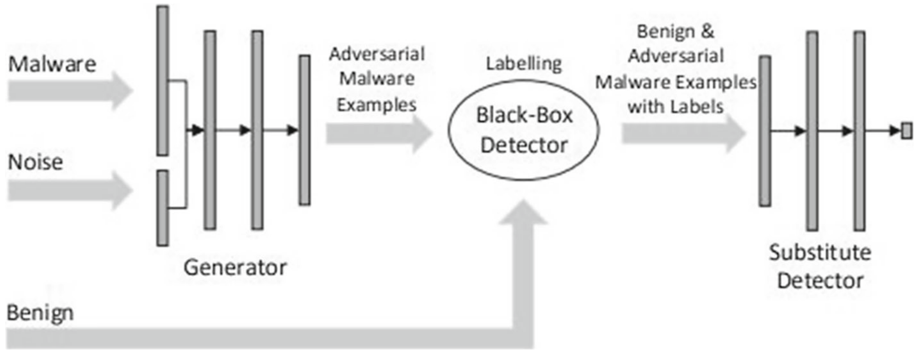

Malware
Noise
Generator
Benign
Adversarial Malware Examples
Labelling
Black-Box Detector
Benign & Adversarial Malware Examples with Labels
Substitute Detector

Fig. 5 MalGAN architecture. Source: Hu and Tan (2017)

(Hitaj et al. 2017) trained a generative adversarial network on over 1 million passwords. The password generating generative adversarial network (*PassGAN*) was then able to generate fake passwords which matched 51–73% passwords in a test set.

GANs have also been used with biometric fingerprint data, researchers trained a GAN in conjunction with stochastic search to generate "Masterprints" that can be used to spoof biometric authentication systems (Bontrager et al. 2017). The results show that on a commercial fingerprint system, the masterprint matches to 22% of all users in a strict security setting, and 75% of all users at a looser security setting.

Generating adversarial examples is one area that has always inspired discussions of the potential risk of security systems and adversarial studies. Dating as far back as Goodfellow's work (Goodfellow et al. 2014b), and more recently in (Kurakin et al. 2016; Elsayed et al. 2018), research from these works have shown that physical systems with classifiers such as an image classifier, will misclassify generated adversarial images as high as 98% of the time. By introducing adversarial perturbations (Kos et al. 2018) to the original image, the researchers were able to fool classifiers into classifying adversarial images incorrectly. Furthermore, there are interesting works such as (Hayes et al. 2019), that show that the ability of GANs to generate adversarial perturbed data can be highly specialized to a particular dataset or domain.

On the subject of malware detection, malware authors often design malware without having much detailed knowledge of the architecture of Malware detectors.

Researchers in Hu and Tan (2017) have designed *MalGAN* (Fig. 5), a GAN based malware generator specifically made for such Blackbox attacks where only the knowledge of the particular input features which are authorized by the malware detector are required. The authors; Hu and Tan (2017) use the gradient based approach to generate adversarial android malware examples. The premise in such an application as seen in Hu and Tan (2017) is that GANs can generate similar and realistic features that can bypass a malware detector and thus fool the system to believe that such a feature is not adversarial.

3.3 Limitations for GANs in cybersecurity application

When using GANs to generate adversarial samples for training the type of systems discussed in Sect. 3.1, the task of considering all potential adversarial samples is a nontrivial task. On their own, GANs do not consider all possible adversaries in a class but rather a small number of adversarial instances are optimized for training (Huang et al. 2018; Dziugaite et al. 2015).

Herein lies a big limitation for training GANs for adversarial detectors in cybersecurity system applications. System designers must engineer mechanisms for considering larger sample of class instances when training GANs.

One major problem peculiar to the Deep Neural Network (DNN) is the interpretability of the results obtained and the black-box nature of these systems. GANs, being a specialized implementation of DNNs pose these problems to system designers. This is one area were rule based systems and other forms of cybersecurity software hold an advantage over neural network based systems. Despite this setback of interpretability, GANs offer robustness and scalability that is often lacking in rule based software.

The difficulties with interpreting and understanding neural networks such as GANs, places a constraint on the ways in which these systems can be modified. Also, the process of training large scale GANs is not a simple and trivial task. System designers for malware detection systems often need to keep up with the rapid changes that exist within the cybersecurity world. Malware authors are rapidly improving upon their systems and malware detection systems will need to keep up with these changes. Given the inherent difficulties of modifying and training an efficient GAN-based security system, designers may find it difficult to keep up with rapidly evolving malware adversarial systems.

There are other limitations with respect to evaluating the performance of GANs as well empirically appraising the quality of generated data. These limitations are quite noteworthy and have been discussed previously in Sect. 2.

4 Discussion

The Generative Adversarial Network is a relatively novel model dating back just four years, so it is safe to say that the landscape of possible applications is still very much open for exploration. Yet within the context of cybersecurity applications for the GAN, one can clearly observe two distinct paths taken. The first path involves research where the ability to detect adversarial attacks it strengthened using generative adversarial networks. The second path involves research that makes use of the GAN to create adversarial systems or adversarial data. Research reviews from these two paths shows that there is promise in future research in cybersecurity involving GANs. However, there are a few setbacks for cybersecurity application of GANs. Much work is needed in ensuring that GANs are able to generate samples from a larger sample space that considers all possible adversarial classes. There is also the need to be able to empirically analyze the training performance and results of the GAN.

The future success of GAN-based cybersecurity systems will be determined by the interpretability of these neural networks. However, there is the potential of integrating GANs with other forms of rule based cybersecurity software. The GANs speed and robustness coupled with the interpretability of these rule-based systems may enable the design of more relatively functional systems. There is much room for research in GAN-based cybersecurity systems involving other types of data such binary or audio-visual data. Finally, the training process for large scale GANs must be made easier if we are to see rapid integration with cybersecurity systems.

5 Conclusion

Generative Adversarial Networks are capable of learning a density function that describes some data sample and can use this estimated density function to generate realistic samples. GANs achieve this by training in a minimax game, two neural networks where the Nash equilibrium point of the game indicates that the system has accurately estimated the density function. GANs are not always perfect and are notorious for being difficult to train, tune and evaluate. Researchers have developed novel architectures that strive to solve these problems. In any case, GANs have provided the state-of-the-art in many areas of application and is currently one of the most successful deep generative models. Applications in cybersecurity with GANs are just starting to gain strong footing and we can observe two major approaches in which GANs can be applied. One approach is to use the GAN in the process of improving an adversarial detector's generalization ability by augmenting the training data with fake samples generated by the GAN. The second notable approach is to use GANs to create adversarial systems that can learn what authorized features look like and can thus generate fake data that has a higher chance of fooling a security system.

References

Abadi M, Andersen DG (2016) Learning to protect communications with adversarial neural cryptography. arXiv preprint arXiv:1610.06918

Anderson HS, Woodbridge J, Filar B (2016) DeepDGA: adversarially-tuned domain generation and detection. In: Proceedings of the 2016 ACM workshop on artificial intelligence and security. ACM, pp 13–21

Apruzzese G, Colajanni M, Ferretti L, Guido A, Marchetti M (2018) On the effectiveness of machine and deep learning for cyber security. In: 2018 10th international conference on cyber conflict (CyCon). IEEE

Arjovsky M, Chintala S, Bottou L (2017) Wasserstein gan. arXiv preprint arXiv:1701.07875

Bengio Y, Thibodeau-Laufer E, Alain G, Yosinski J (2014) Deep generative stochastic networks trainable by backprop. In: ICML'2014

Biggio B, Roli F (2018) Wild patterns: ten years after the rise of adversarial machine learning. Pattern Recogn 84:317–331

Bontrager P, Togelius J, Memon N (2017) Deepmasterprint: generating fingerprints for presentation attacks. https://arxiv.org/abs/1705.07386

Chen X, Duan Y, Houthooft R, Schulman J, Sutskever I, Abbeel P (2016) Infogan: interpretable representation learning by information maximizing generative adversarial nets. In: Advances in neural information processing systems, pp 2172–2180

Denton EL, Chintala S, Fergus R (2015) Deep generative image models using a laplacian pyramid of adversarial networks. In: Advances in neural information processing systems, pp 1486–1494

Dziugaite GK, Roy DM, Ghahramani Z (2015) Training generative neural networks via maximum mean discrepancy optimization. arXiv preprint arXiv:1505.03906

Elsayed GF, Shankar S, Cheung B, Papernot N, Kurakin A, Goodfellow I, Sohl-Dickstein J (2018). Adversarial examples that fool both human and computer vision. arXiv preprint arXiv:1802.08195

Frey BJ, Hinton GE, Dayan P (1996) Does the wake-sleep algorithm produce good density estimators? In: Advances in neural information processing systems, pp 661–667

Goodfellow I (2016) NIPS 2016 tutorial: generative adversarial networks. arXiv preprint arXiv:1701.00160

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp 2672–2680

Goodfellow IJ, Shlens J, Szegedy C (2014) Explaining and harnessing adversarial examples. *arXiv preprint* arXiv:1412.6572

Goodfellow I, Bengio Y, Courville A, Bengio Y (2016) Deep learning, vol 1. MIT press, Cambridge

Grosse K, Papernot N, Manoharan P, Backes M, McDaniel P (2017) Adversarial examples for malware detection. In: European symposium on research in computer security. Springer, Cham, pp 62–79

Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC, (2017) Improved training of wasserstein gans. In: Advances in neural information processing systems, pp 5767–5777

Hayes J, Melis L, Danezis G, De Cristofaro E (2019) LOGAN: membership inference attacks against generative models. Proceedings on Privacy Enhancing Technologies 2019(1):133–152

Higgins I, Matthey L, Pal A, Burgess C, Glorot X, Botvinick M, Mohamed S, Lerchner A (2016) beta-vae: learning basic visual concepts with a constrained variational framework

Hinton GE, Sejnowski TJ (1986) Learning and relearning in Boltzmann machines. Parallel distributed processing: Explorations in the microstructure of cognition 1:282–317

Hitaj B, Gasti P, Ateniese G, Perez-Cruz F (2017) Passgan: a deep learning approach for password guessing. arXiv preprint arXiv:1709.00440

Hu W, Tan Y (2017) Generating adversarial malware examples for black-box attacks based on GAN. arXiv preprint arXiv:1702.05983

Huang C, Kairouz P, Chen X, Sankar L, Rajagopal R (2018) Generative adversarial privacy. arXiv preprint arXiv:1807.05306

Hyvärinen A, Pajunen P (1999) Nonlinear independent component analysis: existence and uniqueness results. Neural Netw 12(3):429–439

Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift

Isola P, Zhu JY, Zhou T, Efros AA (2017) Image-to-image translation with conditional adversarial networks (2016). arXiv preprint arXiv:1611.07004

Kim JY, Bu SJ, Cho SB (2018) Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders. Information Sci 460:83–102

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980

Kingma DP, Salimans T, Welling M (2016) Improving variational inference with inverse autoregressive flow. In: NIPS

Kos J, Fischer I, Song D (2018) Adversarial examples for generative models. In: 2018 IEEE security and privacy workshops (SPW). IEEE, pp 36–42

Kurakin A, Goodfellow I, Bengio S (2016) Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533

Ledig C, Theis L, Huszar F, Caballero J, Aitken AP, Tejani A, Totz J, Wang Z, Shi W (2016) Photo-realistic single image super-resolution using a generative adversarial network. In: CoRR, abs/1609.04802

Li H, Lin Z, Shen X, Brandt J, Hua G (2015) A convolutional neural network cascade for face detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5325–5334

Lin Z, Shi Y, Xue Z (2018) IDSGAN: generative adversarial networks for attack generation against intrusion detection. arXiv preprint arXiv:1809.02077

Lotter W, Kreiman G, Cox D (2016) Deep predictive coding networks for video prediction and unsupervised learning. arXiv preprint arXiv:1605.08104

Malhotra Y (2018) Machine intelligence: AI, machine learning, deep learning & generative adversarial networks: model risk management in operationalizing machine learning for algorithm deployment

Oord AVD, Kalchbrenner N, Kavukcuoglu K (2016) Pixel recurrent neural networks. arXiv preprint arXiv:1601.06759

Radford A, Metz L and Chintala S (2015) Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint* arXiv:1511.06434

Rezende DJ, Mohamed S, Wierstra D (2014) Stochastic backpropagation and approximate inference in deep generative models. In: ICML'2014. Preprint: arXiv:1401.4082

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 2226–2234

Shi H, Dong J, Wang W, Qian Y, Zhang X (2017) Ssgan: secure steganography based on generative adversarial networks. In: Pacific Rim conference on multimedia. Springer, Cham, pp 534–544

Springenberg JT, Dosovitskiy A, Brox T, Riedmiller M (2015) Striving for simplicity: the all convolutional net. In: ICLR

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

Tang W, Tan S, Li B, Huang J (2017) Automatic steganographic distortion learning using a generative adversarial network. IEEE Signal Process Lett 24(10):1547–1551

Theis L, Oord AVD, Bethge M (2015) A note on the evaluation of generative models. arXiv preprint arXiv:1511.01844

Van Den Oord A, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Kavukcuoglu K (2016) Wavenet: a generative model for raw audio. CoRR abs/1609.03499

Yin C, Zhu Y, Liu S, Fei J, Zhang H (2018) An enhancing framework for botnet detection using generative adversarial networks. In: 2018 international conference on artificial intelligence and big data (ICAIBD). IEEE

Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: European conference on computer vision. Springer, Cham, pp 818–833

Zeiler MD, Krishnan D, Taylor GW, Fergus R (2010) Deconvolutional networks

Zhu J-Y, Krähenbühl P, Shechtman E, Efros AA (2016) Generative visual manipulation on the natural image manifold. In: European conference on computer vision. Springer, pp 597–613

Zügner D, Akbarnejad A, Günnemann S (2018) Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. ACM, pp 2847–2856

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations