Evolutive Adversarially-Trained Bayesian Network Autoencoder for Interpretable Anomaly Detection

Jorge Casajús-Setién jorge.casajus@alumnos.upm.es
Concha Bielza mcbielza@fi.upm.es
Pedro Larrañaga pedro.larranaga@fi.upm.es
Universidad Politécnica de Madrid, ETS de Ingenieros Informáticos, Boadilla del Monte, 28660

Abstract

Semi-supervised detection of outliers with only positive and unlabeled data, which is among the most frequent forms of the anomaly detection (AD) problem in real scenarios, requires for a model to capture the normal behaviour of data from a training set exclusively comprised of normal-labelled data, so new unseen data can be afterwards compared to the induced notion of normality to be flagged -or not- as anomalous. In modelling a certain pattern of behaviour, generative models such as generative-adversarial networks (GANs) have proved to have great performance. Thus, numerous AD algorithms with GANs at its core have been proposed, most of them powered by deep neural networks and relying on an autoencoder for the AD task. In the present work, a novel approach to semi-supervised AD with Bayesian networks using generative-adversarial training and an evolutive strategy is proposed, which aims to palliate the intrinsic lack of interpretability of deep neural networks. The proposed model is tested on a real-world AD problem in cybersecurity.

Keywords: Bayesian networks, Generative models, Adversarial training, Evolutive algorithms, Autoencoder, Anomaly detection, Cybersecurity.

1. Introduction

Detecting and properly flagging previously unseen anomalies, understood as patterns in data that do not conform to expected behaviour, makes for a central problem in several research areas (Chandola et al., 2009). In real-life domains, anomalies might appear in data for numerous reasons. Particularly, we will demonstrate later on how anomaly detection can be used in cybersecurity to spot malicious network traffic containing cyber-attacks, proving that anomalous samples are usually worth of further investigation for data analysts.

Anomaly detection (AD) can be regarded as a classification task, with its goal being the correct assignment of a binary category: 0 (normal) or 1 (anomalous) to a given sample. AD usually involves working with unlabelled data, which narrows the set of techniques feasible for such use to those that belong to the category of unsupervised learning. A frequent scenario, and the one that serves as the main framework all along this paper, takes place when the spotting of anomalies must be done by comparison with a previously learnt notion of normality induced from data labelled as normal. This task falls under the semi-supervised learning from only positive labelled data domain, since it requires data previously marked as normal, but does not need anomalous samples during the learning phase of the model (Madhuri and Rani, 2018). Formally, given a dataset $\mathcal{S}$ containing a series of samples $\mathbf{x} \in \mathbf{X}$ (the data space), which are considered to be representative of the normal behaviour and

are thus normal-labelled, the problem can be defined as the optimization of the parameters $\boldsymbol{\theta}$ of a model that learns the distribution of the normal samples in $\mathcal{S}$, $p_{\mathbf{X}}$, and is able to output an anomaly score $A(\mathbf{x})$ for new test samples, where larger scores indicate anomalous samples. Setting a convenient threshold, ϕ, allows for a separation of the test samples in those considered normal ($A(\mathbf{x}) \leq \phi$) and those considered anomalies ($A(\mathbf{x}) > \phi$).

Generative algorithms, on the other hand, pursue the main goal of modelling complex distributions of data, which suggest they can be used in AD tasks (Mattia et al., 2019) by first learning the distribution of the normal-labelled data and then checking whether a given sample complies or not with the learnt behaviour. Among the most widely used generative algorithms, generative adversarial networks (GANs) stand out as the best-performing ones (Deecke et al., 2018). However, adversarial training ideas are often implemented over deep neural networks (DNNs), which have some fundamental limitations regarding interpretability and explainability stemming from their complex architecture.

In this paper, a novel approach to semi-supervised anomaly detection from normal-labelled data using Bayesian networks is proposed. The conditional probability distributions (CPDs), which are the parameters of the model, as well as the directed acyclic graph structure are learnt via a generative-adversarial training, following an evolutive strategy.

The AD task is carried out by using a Bayesian autoencoder, hence the name of the model: Bayesian evolutive adversarial autoencoder (BEAA). The proposed architecture is afterwards applied to a real world problem in cybersecurity, obtaining competitive performance compared with other state-of-the-art methods. Hereafter the paper is organized as follows: In Section 2, AD using generative models is discussed. Section 3 develops around Bayesian anomaly detection and Bayesian generative models, and in Section 4, the proposed model for BEAA is described, with both its learning algorithm and its AD method being thoroughly detailed. Cybersecurity experiments are analyzed in Section 5, and finally, Section 6 includes a general discussion of the work and future development.

2. Anomaly Detection with Generative Adversarial Models

AD using generative models starts by capturing the underlying distribution of data considered to be normal in a certain domain. In doing so, the generative model becomes able to generate new samples conforming to it.

GANs (Goodfellow et al., 2014a) were introduced by proposing a new training procedure for DNNs based on competition against a discriminator agent, which takes the form of another DNN. The generator network tries to represent, using its parameters $\boldsymbol{\theta}_G$, a mapping $G(\mathbf{z};\boldsymbol{\theta}_G)$ from an m-dimensional space of input noise variables $\mathbf{Z}$ to the output n-dimensional space $\mathbf{X}$, which contains the data samples described by their n attribute variables. The noise vector space where the noise vectors $\mathbf{z} = (z_1, ..., z_m)$ lie is referred to as *latent space*, whereas the space in which the samples $\mathbf{x} = (x_1, ..., x_n)$ lie is called *data space*, with the particularity that the latent space is always of lower dimensionality than the data space ($m < n$). The discriminator network encodes within its parameters $\boldsymbol{\theta}_D$ the function $D(\mathbf{x};\boldsymbol{\theta}_D)$, which takes a sample $\mathbf{x}$ for input and outputs a scalar that represents the probability that $\mathbf{x}$ is a real sample, instead of a fake one, generated by applying $G(\mathbf{z};\boldsymbol{\theta}_G)$ over a random noise vector $\mathbf{z}$ acting as a seed. Both generative and discriminative networks are trained in a two-part minmax game, with D aiming to maximize the probability of assigning the correct label to

a given generated or real sample and G minimizing the chances of D guessing a generated sample as such. Mathematically:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}}[\log D(\mathbf{x};\boldsymbol{\theta}_D)] + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z};\boldsymbol{\theta}_G))].$$

The most basic training algorithm for generative-adversarial neural networks, as presented in Goodfellow et al. (2014a) is detailed in Algorithm 1.

Algorithm 1 Minibatch stochastic gradient descent training for generative adversarial neural networks (Goodfellow et al., 2014a)

for number of training iterations **do**

1. Sample minibatch of N noise samples $\{\mathbf{z}^{(1)}, ..., \mathbf{z}^{(N)}\}$
2. Sample minibatch of N real samples $\{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(N)}\}$
3. Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{N} \sum_{i=1}^{N} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$$

4. Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{N} \sum_{i=1}^{N} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

end for

After adversarially training a discriminator and a generator over a dataset $\mathcal{S}$ of normal-labelled samples, the discriminator can be considered as a rudimentary anomaly detector by using its output as the anomaly score for the categorization of new samples. This method has the main drawback that its behaviour depends heavily on how the training concluded:

- If D was able to correctly flag most of the fake and real samples by the time the training process ended, then the output $D(\mathbf{x};\boldsymbol{\theta}_D)$ for new samples should be closer to 0 for anomalous ones, since the discriminator will know they do not come from the distribution of normal samples, and closer to 1 for normal ones, indicating that the discriminator believes they follow the same distribution as the training examples.
- If G was managing to deceive the discriminator a significant fraction of times when the training stopped (this is, the generator captured accurately the underlying distribution of normal data), then the output $D(\mathbf{x};\boldsymbol{\theta}_D)$ for new samples can reach a point where anomalous samples receive a score closer to 1, since the discriminator would have only been trained on fake examples that resemble very closely the normal-labelled data.

The discriminator is then, though perfectly capable of distinguishing real from forged data, not prepared to deal with samples that differ from the training ones. In order to make more robust anomaly detectors with GANs, other authors have explored the usage of the

generator network for this task. In this line of work, there are two studies which we used as a reference point in AD with generative architectures to build our model upon.

The AnoGAN architecture (Schlegl et al., 2017) emerged as an AD model based on standard GANs trained only on normal (non-anomalous) samples, allowing the generator to synthesize realistic normal samples by the mapping $G(\mathbf{z}; \boldsymbol{\theta}_G) : \mathbf{z} \rightarrow \mathbf{x}$. Such function can also be used to find the inverse mapping to trace a new sample back to its representation in the latent space, $E(\mathbf{x}) : \mathbf{x} \rightarrow \mathbf{z}$. In order to find the $\mathbf{z}$ that leads to the $G(\mathbf{z})$ that is closer to the original $\mathbf{x}$, the coordinates of $\mathbf{z}$ in the latent space are found via an iterative process by minimization of the residual (reconstruction) loss $\mathcal{L}_R(\mathbf{z}_\gamma)$ and the discrimination loss $\mathcal{L}_D(\mathbf{z}_\gamma)$ for each step $\gamma = 1, ..., \Gamma$:

$$\mathcal{L}_R(\mathbf{z}_\gamma; \boldsymbol{\theta}_G) = ||\mathbf{x} - G(\mathbf{z}_\gamma; \boldsymbol{\theta}_G)|| \qquad \text{and} \qquad \mathcal{L}_D(\mathbf{z}_\gamma; \boldsymbol{\theta}_G, \boldsymbol{\theta}_D) = \sigma[D(G(\mathbf{z}_\gamma; \boldsymbol{\theta}_G); \boldsymbol{\theta}_D), 1],$$

where $\sigma(x, \alpha)$ stands for the sigmoid cross-entropy computed over x targetting its class α. The first loss function ensures the similarity between the generated and original samples, while the second loss enforces the generated sample to lie on the learnt representation of the data space from the normal-labelled training samples. The overall loss function is built as a weighted sum of both components (with $\lambda \in [0, 1]$):

$$\mathcal{L}(\mathbf{z}_\gamma; \boldsymbol{\theta}_G, \boldsymbol{\theta}_D) = (1 - \lambda)\mathcal{L}_R(\mathbf{z}_\gamma; \boldsymbol{\theta}_G) + \lambda\mathcal{L}_D(\mathbf{z}_\gamma; \boldsymbol{\theta}_G, \boldsymbol{\theta}_D).$$

Ultimately, the detection of anomalies is performed using as the anomaly score $A(\mathbf{x}) = (1 - \lambda)R(\mathbf{x}) + \lambda D(\mathbf{x})$, where $R(\mathbf{x})$ and $D(\mathbf{x})$ are the residual score and the discrimination score, defined as the residual and discrimination loss computed at the last iterative step of the mapping of the sample to the latent space, $\mathcal{L}_R(\mathbf{z}_\Gamma)$ and $\mathcal{L}_D(\mathbf{z}_\Gamma)$, respectively. The idea behind this anomaly score is that, as the generator is trained over normal examples, an anomalous one will be poorly reconstructed, as the generator will have a hard time trying to find a point in the latent space that both maps to an accurate reconstruction of the original sample and lies in the learnt representation of the data space.

Another AD framework using GANs and normal-labelled data is the GANomaly model (Akcay et al., 2018). At the core of its architecture lies an autoencoder acting as the generative submodel. An autencoder is a special case of encoder-decoder model for which the inputs and outputs belong to the same domain, in this case, the data space. The autoencoder (A) in the GANomaly structure consists of two neural networks: an encoder, A_E, which reads an input sample $\mathbf{x}$ and outputs a compressed representation of it ($\mathbf{z}$) pertaining to the latent space, and a decoder, A_D, which accepts a noise vector $\mathbf{z}$ for an input and upscales such vector back to the data space. Combining these two networks forming a "bowtie" shape (the output of A_E is the input for A_D), a special type of generator is constructed: A gets for an input a sample $\mathbf{x}$ and outputs a reconstruction $\mathbf{x}'$ of it, which is formed by projecting the original sample onto its variables in the latent space and then reconstructing its image in the data space. The GANomaly structure, depicted in Figure 1, also comprises an aditional encoder network, E, which downscales $\mathbf{x}'$ to the latent space once again, returning $\mathbf{z}' = E(\mathbf{x}')$, and a discriminator network D, whose goal is to classify $\mathbf{x}$ and its autoencoded reconstruction $\mathbf{x}'$ as real or fake. The training of the model is carried out by optimizing not only the discrimination and reconstruction losses (in this work these are referred to as adversarial loss and contextual loss, respectively, but they are

formulated in an analogous fashion), but also an encoder loss to minimize the difference between the encoded features of the input ($\mathbf{z} = A_E(\mathbf{x})$) and those of the reconstructed sample, $\mathbf{z}' = E(A_D(A_E(\mathbf{x}))) = E(A(\mathbf{x}))$. It differs from the AnoGAN architecture in the fact that the encoder is itself a neural network trained conjointly with the generative and the discriminative model. For the AD task, the encoder loss is used as an anomaly score, thus, for a test sample $\hat{\mathbf{x}}$:

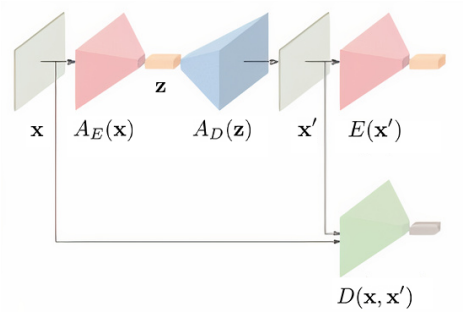


Figure 1: GANomaly structure (Akcay et al., 2018).

$$A(\hat{\mathbf{x}}) = \|A_E(\hat{\mathbf{x}}) - E(A(\hat{\mathbf{x}}))\|.$$

3. Bayesian Anomaly Detection and Generative Models

Quoting again the work of Goodfellow et al. on GANs: "The adversarial modeling framework is most straightforward to apply when the models are both multilayer perceptrons". All of the previously discussed architectures are powered by multilayer perceptrons (namely, neural networks), being these the mathematical paradigms that encode the functions needed for the general approach to work. Neural networks are, though easy to train in an adversarial manner, naturally subject to a series of limitations regarding their lack of interpretability. There are three main reasons why interpretability should be considered as a design objective when developing a machine learning model (Barredo Arrieta et al., 2020), namely:

- It allows for a better detection of biases in the training dataset, enhancing imparciality.
- Interpretability assists in highlighting potential adversarial weaknesses of the model. Adversarial samples are specialized inputs created with the purpose of confusing a model, forcing misclassification (Goodfellow et al., 2014b). This is a particularly serious problem in cybersecurity, since adversarial weaknesses open a way for malicious traffic to fly under the radar of an intrusion detection system based on AD.
- It can help developers and operators to understand how and which variables affect the output of a model, so it can be used to guarantee that the model reasoning follows a meaningful underlying causality, if it existed in the training data.

Bayesian networks (BNs) stand among the most interpretable machine learning models (Mihaljević et al., 2021), but are usually related to lower performance than black-box

models, as deep neural networks (DNNs), specially when facing high dimensional problems. This is the reason why in this work we bring a new generative-adversarial approach based on BNs that aims to be a hybrid AD solution, which provides better accuracy than BNs while retaining part of their inherent interpretability, justifying its use against a pure DNN-based GAN approach in contexts that require interpretability, such as cybersecurity.

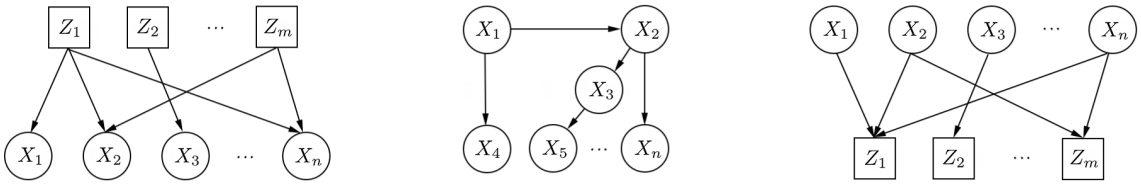
AD using a single BN is easy to perform once the structure (a directed acyclic graph) and parameters of the model, in this case the CPDs, are learnt. It is carried out using the log-likelihood of a sample as a measure of how anomalous it is, and it requires for a threshold to be set for the purpose of separating normal samples from anomalies (Atienza et al., 2021a). In order to construct a GAN-based AD system using BNs, one must first figure out how to adversarially train a Bayesian discriminator-generator couple. Some of the most closely related works are those of Li et al. (2018) on graphical GANs and Ding et al. (2021) regarding GANs with conditional independence graphs.

The first of the two aforementioned works stablishes a generative modelling framework for structured data featuring a generator that has an associated directed acyclic graph (DAG), so samples can be efficiently generated via ancestral sampling. Still, albeit the dependencies are represented by a DAG, the dependency functions among the variables are parametrized as DNNs. It is precisely in the implicit treatment of probabilities, by defining a stochastic process that aims to draw samples according to an unspecified distribution, that these graphical GANs most closely resemble our proposed algorithm (see Section 4). The parameters of the model are found by gradient descent in an adversarial manner, trying to maximize with respect to the discriminator parameters and minimize with respect to the generator ones the divergence between the probability distributions of both a generative and a recognition submodel. The latter work is also focused on providing GANs with a graphical structure, turning them into a model-based design, with the difference that their probabilistic approach computes the optimal parameters by minimization of the upper bound of a general divergence measure using local discriminators and consequently improving computational efficiency.

4. BEAA: Model Structure, Learning Algorithm and AD Mechanism

In our proposal, not only the generative and discriminative models are built upon a probabilistic graphical structure, but they straightforward are BNs. Both the CPDs and the graph structure are iteratively and adversarially learnt in a minimization problem following an evolutive strategy. The AD task is performed by an autoencoder system, hence the name of the framework. The model has been implemented for posterior experimentation on Python using the PyBNesian library (Atienza et al., 2022). Thus, its advantages and limitations have been decisive in the design of a compatible structure and the development of the learning and AD algorithms. One of the key aspects of the library is that nodes that accept evidence must have no parents, and are referred to as interface nodes. In its PyBNesian-compatible form, the model integrates three BNs:

- A generator BN, $\mathcal{G}$, whose parameters are learnt in an evolutive procedure by adversarial competition against the discriminator. Unlike the original GAN training schema, which alternatively optimizes the generator and the discriminator in an iterative way, here the discriminator is not developed conjointly with the generator, but



(a) $\mathcal{G}$ structure (b) $\mathcal{D}$ structure (c) $\mathcal{R}$ structure

Figure 2: The three components of the BEAA for a generic example case with n attribute nodes (circles) and m noise nodes (squares).

the generator is trained by confronting a global and static discriminator. The graph of the generator network is devised to comply with the following constraints:

  - Latent variables (noise nodes) must be root nodes, having no parents, as these will be the interface nodes for the BN.
  - Data space variables (attribute nodes) must be leaf nodes, having no children.

  In addition, we considered the model to be free of arcs from one noise variable to another, thus containing all of the information about dependencies of attributes in the arcs connecting noise nodes to attribute nodes. This way, the generator network is suitable to perform ancestral sampling of forged examples following its underlying distribution from a set of noise variables acting as seeds.

- A discriminator network, $\mathcal{D}$, which is only used during the training phase of the model. This one is a standard Bayesian network learnt directly from the training dataset.
- An inverse generator network, $\mathcal{R}$, which is learnt from $\mathcal{G}$ and represents its inverse function, returning the expected noise from an input sample. The constraints over the inverse generator are the same applied over the generator network, but reversed:
  - Noise nodes must be leaf nodes, having no children.
  - Attribute nodes must be root nodes, having no parents, as they will be interface nodes. Besides, arcs between attribute nodes are forbidden.

The DAG structure for each of the components is illustrated in Figure 2. The decision of using two separate networks for generation of data from noise (decodification) and prediction of noise from data (codification), as well as the structure used for both, was made taking into account that the PyBNesian library only supports evidence-driven ancestral sampling of data by instantiating root nodes. In the practical implementation, all of the three BNs of the BEAA are defined as conditional linear Gaussian BNs, with each CPD being:

$$f(x_i|\mathbf{e}) = \mathcal{N}(x_i; \beta_0 + \sum_{j=1}^{k} \beta_{ij} e_j, \sigma^2),$$

where x_i denotes the variable for which the CPD is defined, $\mathbf{e}$ is the k-dimensional vector containing the instantiation of the parents of the corresponding variable used as evidence,

$\boldsymbol{\beta} = (\beta_0, \beta_1, ..., \beta_k)$ is the vector of parameters modelling the dependencies between nodes and σ^2 stands for the variance of the conditional Gaussian (Koller and Friedman, 2009). The choice of linear Gaussian BNs was made out of practicity, since these are readily available in the PyBNesian library, though the model can also be implemented for any kind of BNs, such as conditional KDE networks or semiparametric Bayesian networks (Atienza et al., 2021b), also included in the package. Several learning algorithms have been designed for the model, but the most basic and PyBNesian compliant form of it goes by the following guidelines:

1. In the first instance, the discriminator $\mathcal{D}$ is adjusted to fit the initial distribution of the training data $\mathcal{S}$, comprised exclusively of positive-labelled samples.

2. Following, the generator network $\mathcal{G}$ is initialized. This is carried out by fully connecting each noise node to each attribute node. The parameters of the initial probability distributions are randomized and act as the starting seed for the training procedure.

3. The optimal parameters for the generator are found in an iterative process starting from the randomized initialization of $\mathcal{G}$. For each iteration, a batch of samples is generated according to the distribution encoded in $\mathcal{G}$ using Gaussian noise as evidence for the noise nodes and following ancestral sampling. The log-likelihood according to $\mathcal{D}$ (and thus, only regarding attribute nodes) of each of the generated samples is calculated, so that a fraction of the best-scoring ones is chosen as the training set for the generator that will be used in the next iteration. This is an evolutive process that reminds of the estimation of distribution algorithms (Larrañaga and Lozano, 2002), since each generation of samples is estimated from a simulation of a probability distribution inferred by the selected samples of the previous generation. This process is repeated until convergence, i.e., until a maximum number of epochs (training iterations) is reached or until a certain log-likelihood threshold is met.

4. Once the generator has learned its definitive parameters, the inverse generator must be learned. At a practical level, this has been implemented by generating a final batch of samples according to $\mathcal{G}$ and learning the parameters of $\mathcal{R}$ from it. The set of learnt arcs from the attribute nodes to the noise ones and the CPDs that describe such dependencies constitute the interpretable element that gives the BEAA an upper hand against standard GANs in certain contexts.

A pseudocode version of the algorithm is featured in Algorithm 2. As for the technical learning module implementation, all of the used structure learning and parameter estimation algorithms belong to the PyBNesian library. For all three BNs, experiments have been conducted with the PC structure discovery algorithm (Spirtes et al., 1993), using both mutual information and linear correlation for the conditional independence tests. Parameters are always found by maximum likelihood estimation.

A more sophisticated -though less time-efficient- version of the algorithm, including a greedy search heuristic for the optimal number of noise nodes, has also been implemented, repeating the learning procedure until convergence for an incremental number of noise variables up to the point where the mean value of the log-likelihood computed over the last batch of samples does not improve by adding another noise node.

Algorithm 2 Adversarial-evolutive learning algorithm for a Bayesian autoencoder with a fixed number m of noise nodes and n attribute nodes

Learn discriminator $\mathcal{D}$ with n nodes from normal training data
Initialize generator $\mathcal{G}$ with m noise nodes and n attribute nodes
for number of noise nodes in $\mathcal{G}$ **do**

1. Generate n random coefficients β_{ij} for noise node i, where $j = 1, ..., n$
2. Fully connect noise node i to each attribute node j with coefficient β_{ij}

end for
repeat

1. Sample batch of $N_{\text{generation}}$ samples from $\mathcal{G}$
2. Compute log-likelihood of samples according to $\mathcal{D}$
3. Select $N_{\text{selection}}$ best scoring samples
4. Update the generator by re-learning its structure and parameters according to the selected minibatch of samples

until convergence or maximum number of epochs
Sample batch of $N_{\mathcal{R}}$ samples from $\mathcal{G}$
Learn structure and parameters of $\mathcal{R}$ from generated samples

The discriminator and generator model, though useful on their own for Bayesian generation of forged coherent samples, are just a means to an end in this work, since the main pursue was to build an AD system, and it will be the inverse generator $\mathcal{R}$ the element carrying this task out, for what we present two methods, both of them based on outlier classification according to an anomaly score.

The first method recognizes abnormalities in data by looking at the Euclidean module of the noise vector that $\mathcal{R}$ calculates for a given sample. The intuition behind this idea is built upon the fact that, after training, each noise node will encode a statistical relationship between a subset of the attribute variables that, provided training and test data were previously normalized, should diverge from 0 when anomalies are present. The second method is a straightforward adaptation of the reconstruction error mentioned in Section 2. By calculating the module of the difference between the original and reconstructed sample, an estimation of how poor the reconstruction is, and thus, of how anomalous the sample is, can be obtained. Both methods provide interpretation insight, since the relationship(s) responsible for the anomaly will be highlighted by looking at the most deviated noise direction(s). In the experiments presented in Section 5, a combination by multiplication of both measures, referred to as the combined measure, was also put to test for AD.

5. Experimental Results

The BEAA implementation used for the following experiments can be found submitted to GitHub. All of experiments were conducted on the UNSW-NB15 public cybersecurity dataset Moustafa (2017), containing 2,540,044 records of network traffic and including 9 types of different attacks, which will all be merged into the anomalous class for this prob-

lem, summing up to 321,283 anomalies. Some of the variables present in the dataset are synthetically calculated through packet-level data processing, and were dumped in order to test the attack detection capabilities of the system exclusively at the network traffic level, leaving a total of 24 variables to be taken into account. It has been experimentally demonstrated that the training scheme consistently improves the log-likelihood of the generated samples. Two cases of evolution of the mean log-likelihood according to $\mathcal{D}$ of a batch samples produced by $\mathcal{G}$ are illustrated in Figure 3. Here, quick convergence towards more realistic samples is achieved, even though random initialization of noise CPDs causes the $\mathcal{D}$-likelihood of the initial generated samples to diverge towards negative infinity. For comparision, $\mathcal{D}$ reports a value of 5.25 for the log-likelihood averaged over the subset of normal-labelled test examples.

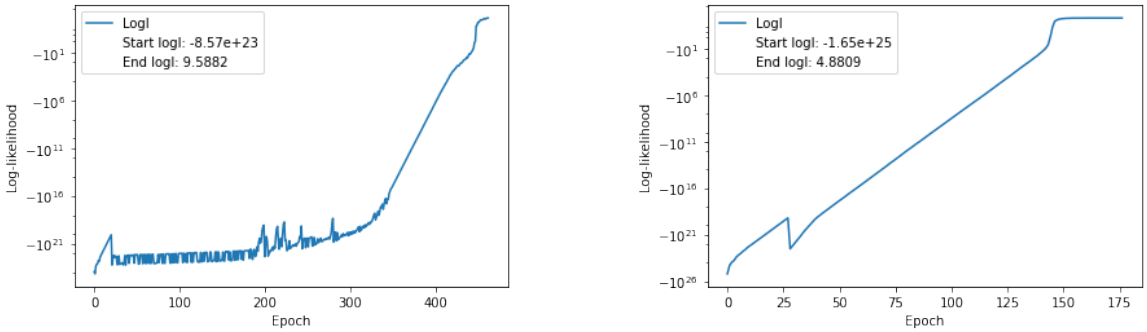


Figure 3: Mean log-likelihood evolution for two training cases on the UNSW-NB15 dataset. Logarithmic scale is used for visualization purposes.

It has also been proved that an improvement on the expected log-likelihood of generated samples is directly correlated to a better separation between the anomalies and the normally behaved samples. However, overtraining of the model can lead to all of the multivariate linear conditional Gaussian distributions collapsing into univariate independent Gaussian distributions, what ultimately entails a dramatic decrease of the AD power of the model. This, together with to-be-resolved instabilities in the learning algorithm, has rendered impossible to prove in statistical tests that our model consistently achieves higher accuracy on detection of anomalies in the UNSW-NB15 dataset compared with a simple BN working as an anomaly detector, which deterministically achieves an area under the ROC curve score of 0.9805. Still, the BEAA has scored significantly over this threshold multiple times under the same setup, as illustrated in Figure 4.

6. Discussion and Future Lines of Work

Overall, the proposed model has proved useful in stretching the performance of a BN applied to anomaly detection at the cost of some interpretability. Despite the BEAA not having yet reached its full potential, specially in terms of training stability, it already shows a promising advance in Bayesian generative models, providing, as demonstrated, interpretable AD.

This one is an ongoing work, though. As such, the idea we have deemed to have the greatest potential incorporates at each training iteration a reconfiguration of the points of

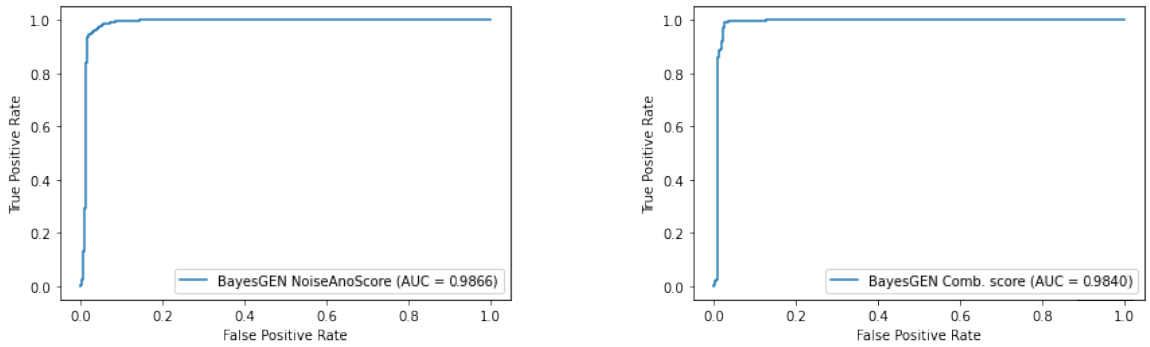


Figure 4: Area under the ROC curve for two learnt models. The models in the left and right panel use, respectively, the noise module anomaly score and the combined one.

the latent space by reparametrizing their coordinates proportionally to the $\mathcal{D}$-likelihood of each sample. We also consider to be worthy of future research the usability of the proposed generative Bayesian model as a tuneable synthetic data generator. We will also be working on further experimenting with the BEAA model in order to characterize its interpretability properties and exhaustively describe how to adjust its parameters to get optimum results.

Acknowledgments

This project was developed in collaboration with Titanium Industrial Security S.L. as a part of the SLISE (network SLIcing SEcurity for next generation communications) program.

References

S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon. GANomaly: Semi-supervised anomaly detection via adversarial training. In *Asian Conference on Computer Vision*, pages 622–637. Springer, 2018.

D. Atienza, C. Bielza, J. Diaz-Rozo, and P. Larrañaga. Efficient anomaly detection in a laser-surface heat-treatment process via laser-spot tracking. *IEEE/ASME Transactions on Mechatronics*, 26(1):405–415, 2021a.

D. Atienza, C. Bielza, and P. Larrañaga. Semiparametric Bayesian networks. *Information Sciences*, 584:564–582, 2021b.

D. Atienza, C. Bielza, and P. Larrañaga. PyBNesian: An extensible python package for Bayesian networks. *Neurocomputing*, 504:204–209, 2022.

A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41, 07 2009.

L. Deecke, R. A. Vandermeulen, L. Ruff, S. Mandt, and M. Kloft. Image anomaly detection with generative adversarial networks. In *ECML/PKDD*, pages 3–17, 2018.

M. Ding, C. Daskalakis, and S. Feizi. GANs with conditional independence graphs: On subadditivity of probability divergences. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3709–3717. PMLR, 2021.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 3, 2014a.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv 1412.6572*, 2014b.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

P. Larrañaga and J. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Genetic algorithms and evolutionary computation. Springer New York, NY, 2002.

C. Li, M. Welling, J. Zhu, and B. Zhang. Graphical generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

S. Madhuri and M. U. Rani. Anomaly detection techniques causes and issues. *International Journal of Engineering Technology*, 7(3.24):449–453, 2018.

F. D. Mattia, P. Galeone, M. D. Simoni, and E. Ghelfi. A survey on GANs for anomaly detection. *CoRR*, abs/1906.11632, 2019.

B. Mihaljević, C. Bielza, and P. Larrañaga. Bayesian networks for interpretable machine learning and optimization. *Neurocomputing*, 456(C):648–665, 2021.

N. Moustafa. *Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic.* University of New South Wales, Canberra, Australia, 2017.

T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *CoRR*, abs/1703.05921, 2017.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. Lecture Notes in Statistics. Springer New York, NY, 1993.