GANs for Time Series Anomaly Detection

Giacomo Mariotti
giacomo.mariotti@alpha-i.co

Fergus Simpson
fergus.simpson@alpha-i.co

November 23, 2018

1 Introduction

This document describes Alpha-i's work on algorithms for time-series anomaly detection based on Generative Adversarial Networks (GAN). The relative code is available in the dedicated GitHub repository *alpha-i/detective-gan-rick-and-morty*. We first give an introduction to GANs and their use in the context of anomaly detection and then describe the development and benchmarking of multiple GAN-based models for anomaly detection using a publicly available time-series dataset.

1.1 GANs

GANs are architectures comprised of two deep neural networks, competing with each other. These two networks are called the Generator and the Discriminator[1]. The Generator generates new data instances, while the Discriminator evaluates their authenticity, deciding if the instance of data it is processing belongs to the class of training data.

GANs work according to the following steps (see Figure 1) :

- The Generator takes in random numbers and returns fake data
- The generated data is fed into the Discriminator, in addition to a stream of data taken from the training dataset
- The Discriminator takes in both real and fake data and returns predictions of authenticity for each of them.

The system composed of the two neural networks learns through two feedback loops:

- The Discriminator receives feedback from the ground truth of real and fake data
- The Generator receives feedback on how successful it is at fooling the Discriminator

1.1.1 GANs for anomaly detection

GANs can be used as an unsupervised learning for perform anomaly detection, thanks to their ability of abstraction and representation of complex data and, while at the same time learning a discriminative and a generative component.

In fact, one powerful feature provided by GAN-based anomaly detection is the Generator's ability to learn generative models generating detailed realistic data. This can be used when new data is presented to the trained GAN to create a synthetic data sample that closely resembles the input data. If the Generator is able to fully represent the new data, this will be an indication that the new sample belongs to the same manifold of the training data. Should the Generator fail to reproduce the new sample data, the difference between the best synthetic data it could come up with and the target sample data can be used as root cause analysis. This way we can pin down those characteristics of the new data that don't belong to the training set.

In this work, to detect anomalies we use the Discriminator part of the GAN (see Figure 2).

- New data is presented to the Discriminator
- The Discriminator predicts the probability that the new data is anomalous

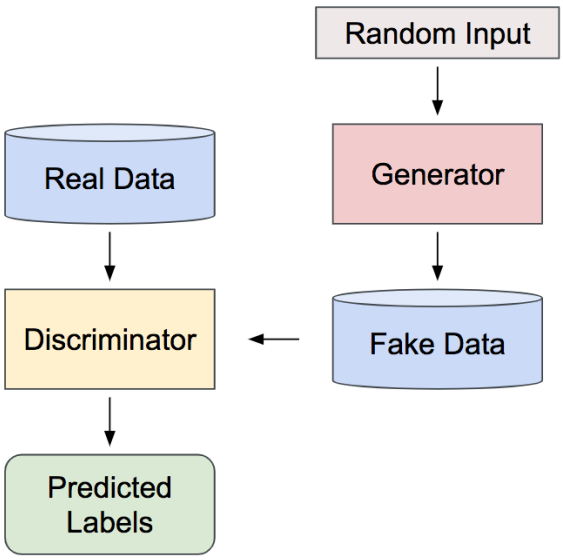

Figure 1: Diagram describing the architecture of Generative Adversarial Networks (GAN).

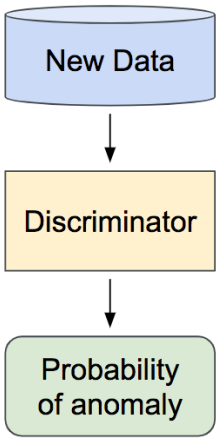

Figure 2: Diagram describing the use of the Discriminator for anomaly detection.

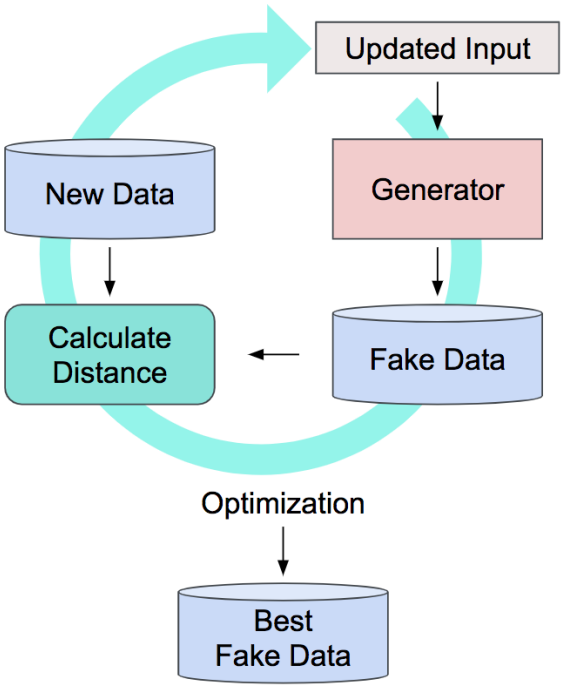

Figure 3: Diagram describing the use of the Generator for Root Cause Analysis.

To identify the root cause of an anomaly we use the Generator part of the GAN (see Figure 3).

- The generated sensor data is compared with the new data being evaluated and a measure of distance is calculated
- An optimization is carried out to find the closest representation to the new data that the Generator can produce
- The difference between real and best fake data highlights the root causes behind the detected anomaly

At the time of writing, the methods described in this section have been extended for improved accuracy and efficiency[3, 7, 9].

2 The Problem

The key objective is to assign a probability that a given time series belongs to a class of known time series, for example corresponding to the nominal behaviour of a sensor.

2.1 Data

To develop the methods for time series anomaly detection and evaluate their performance we use the publicly available dataset, which form part of Kaggle's Seizure Prediction Challenge.

This datasets consists of multivariate time series of Intracranial EEG (iEEG). The data is organized into ten minute EEG clips labeled *Preictal* for pre-seizure data segments (*abnormal* for our purposes), or *Interictal* for non-seizure data segments (*normal* for our purposes). The clips have length $\sim 10^6$ data points.

To make this manageable for a network input, we can split this into 100 sections of 10^4. A small penalty introduced here is we lose some very low frequency information.

2.2 Transformation

It may prove advantageous to study the power spectrum of the time series, rather than the raw output of the sensors, as the detector can more easily identify if an anomalous frequency has

emerged. This is also straightforward to compute using the `np.fft()` routine.

Another means of transforming the data is simply to form a discretised probability density function (pdf) of the segment. This greatly reduces the amount of data per segment, but may make it easy to spot certain kinds of anomalies.

For each method below we shall explore the performance for both raw and transformed signals.

2.3 Normalisation

Real world data is rarely conveniently normalised, so we first apply a degree of normalisation to the data before it enters each model. In this work we use scikit's QuantileTransformer, which ensures a near-uniform probability distribution of the data between 0 and 1.

The same normalisation is applied to the training data as well as the dataset used for testing.

3 Methods

To develop advanced anomaly detection algorithms based on GANs and assess their performance, we introduce a reference method and consider some variations of GAN-based methods. The following subsections describe the various methods considered in this work.

3.1 Method A: SVM

The Support Vector Machine (SVM) is a popular supervised learning method often used for binary classification. Thanks to the notion of hyperplane distance, it allows probabilistic classification with only the "normal" class of data represented in the training set.

SciKitLearn has a build in SVM class designed for single classification problems, which we will use for our benchmarking. One of the main drawbacks of the SVM is that it does not scale well to larger datasets, becoming prohibitively slow when training on a large number of samples.

3.2 Method B: Basic GAN

A GAN topology consists of two networks: a generator and a discriminator.

In our case, the generator learns to generate time series which appear similar to those in the training set. Meanwhile the discriminator learns to distinguish between the real time series and the fake ones produced by the generator. During the training process, each network is trained alternately, gradually learning to surpass the performance of its predecessor.

GANs can often develop issues with stability during training, for example if the discriminator becomes so strong that it cannot improve further, and the generator may struggle to find a way to trick the discriminator.

Training can progress in an unsupervised manner - in other words, we don't have to provide the correct answers, it can just run with the raw data. After a period of training has progressed, the generator can be used to produce fake realisations of the data, and the discriminator is often discarded. For our purposes, the discriminator can be useful in diagnosing whether new data is anomalous or not.

For Method B we select a simple GAN architecture composed of two layers which serves as a first benchmark step into the technology.

3.3 Method C: Bayesian GAN

When the amount of available data is small, the performance of Bayesian GANs often outstrips conventional methods [5]. A publicly available example of a Bayesian GAN algorithm can be found at www.github.com/andrewgordonwilson/bayesgan.

This deep Bayesian network uses a Hamiltonian Monte Carlo method to keep track of the probabilities of the network parameters. This setup has been adapted for usage for anomaly detection.

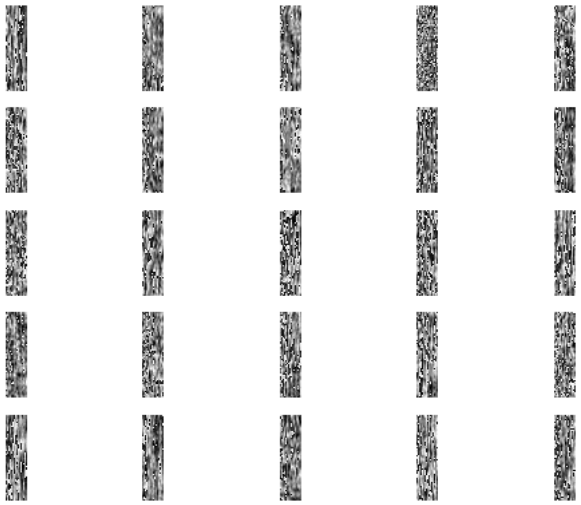

Figure 4: EEG data from a dog.

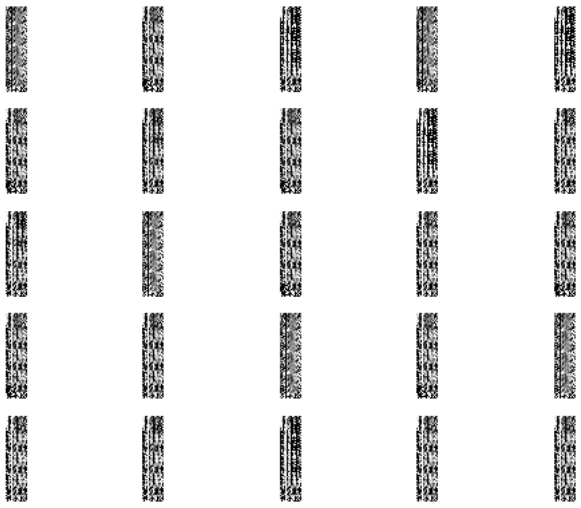

Figure 5: Fake eeg data from a GAN during the early stages of training.

3.4 Method D: Wasserstein GAN

The single most common issue with GANs is mode collapse, where the generator ends up returning always the same output. It is extremely important to keep an eye on how the generator's training progresses, as mode collapse is an irreparable failure in the training process.

To tackle this challenge, we evaluate the Wasserstein GAN (WGAN), which improves the stability of learning and gets rid of the problem of mode collapse by introducing a new cost function using the Wasserstein distance, which has a globally smoother gradient. [2, 4, 6, 8]

4 GANs and EEG data

We train our GANs only with normal (interictal) EEG data of dogs and humans. Then the discriminator component of the network is fed known anomalous data (preictal) as a test of its discriminatory ability.

One could also train with some anomalous data, but we wish very different kinds of anomalies to be detected, so this is not a particularly beneficial approach.

In Figure 4 we can see examples of the true EEG data, while Figure 5 shows some early results from the generator.

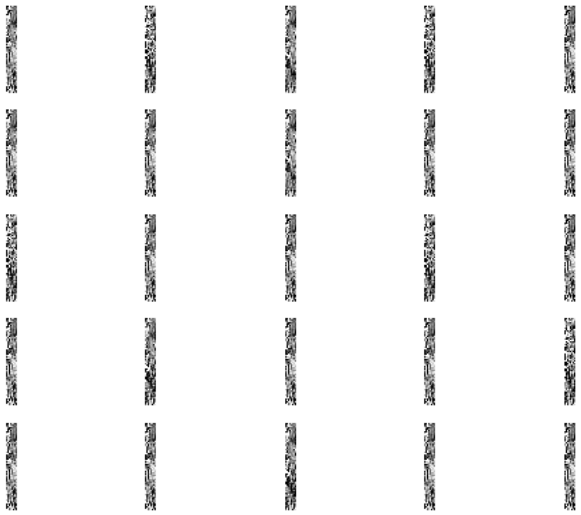

Figure 6: Fake eeg data from bayesGAN after 3 hours of training, spanning 50,000 iterations. There is now less sign of the artificial structure seen in the early phase of training. However there are still signs that the generator is repeating the same characteristics, a common phenomenon known as mode collapse.

5 Results

To evaluate results we train our models on 200 normal (interictal) series. The task is then to identify which of 40 test series are anomalous (20 normal and 20 abnormal, known as preictal, as they correspond to brain activity shortly before a seizure).

A single segment is much too long to be passed into a single network at once. Thus each segment is split into many subsegments, and each subsegment then receives an individual score to indicate whether it is believed to originate from a regular or anomalous source. The series as a whole is then assigned a probability based on the product of the probabilities for each of the n segments:

$$p(Anomaly) = \frac{\Pi_n p(s_n = A)}{\Pi_n p(s_n = A) + \Pi_n p(s_n = R)}$$

where $p(s_n = R) = 1 - p(s_n = A)$ is the probability that the segment originates from a regular series.

5.1 Defining the Kaggle Benchmark

The training set consists solely of normal data (Dog 1 interictals: 1-200). The test set then consists of an equal mix of normal and abnormal data (Dog 1 interictals: 201-220; Dog 1 preictals: 1-20).

The benchmark score we shall use is known as the ROC score, which is a rather ad-hoc quantity but is widely used as a simple metric to quantify the performance of classifiers. Care must be taken in interpreting the uncertainty of the score, which can be crudely estimated by performing multiple independent tests on the same detector.

A total of 504 different users submitted their results to the Kaggle leaderboard. They were able to use labelled preictal (abnormal) examples as part of the training process. The median contestant (rank 252) ended with a score just under 0.6. This is roughly equivalent to a classifier which correctly distinguishes 60% of the segments correctly. We should not expect to do much better than this, due to the hindrance of not using the abnormal data for training.

5.2 Method Performance

For all methods we present the ROC score achieved at model convergence. On the topic of model convergence the following must be noted:

- convergence was easily achieved for the SVM model;

- both the Simple GAN and the Bayesian GAN incurred several training issues like mode collapse, typical of this architecture. However, the Bayesian model showed improved insulation from overfitting issues with respect to the Simple GAN method;
- the Wasserstein GAN showed greater learning stability and robustness to mode collapse issues.

The following table provides a summary of the results.

ROC Score		
	Raw Data	Power Spectrum
SVM	0.68	0.65
Basic GAN	0.59	0.63
Bayesian GAN	0.63	0.68
Wasserstein GAN	0.63	0.74

6 Conclusions

In this work, we developed operational code for time series anomaly detection using GANs, following a step-wise increase in model complexity and implementing the latest methods from the literature.

We incurred the main known issues with simple GAN models, namely overfitting and mode collapse, and addressed them using Bayesian methods and the Wasserstein methodology.

Performance results for the task of anomaly detection on the brainwave dataset showed the following:

- the simple SVM model sets a high bar for the task, achieving a ROC score of ~ 0.68, which would rank 82nd out of 504 competitors on Kaggle. This is remarkable, considering that no abnormal data was used during training (normal and abnormal data are available for training to Kaggle participants).
- to beat the SVM benchmark, considerable model complexity is required using the GAN methodology. However, the introduction of the Wasserstein GAN allowed us to cross the 0.68 ROC level established by the SVM up to a value of 0.74.
- although, the Power Spectrum appears to be detrimental for the SVM model, it proves crucial to push the detection performance of our GAN models. This deserves further investigation, but we believe it's probably due to the different ways the two model classes have of abstracting and representing data.

In conclusion, we believe that we have only started scratching the surface and that GAN-based methods have a huge potential in the context of time-series anomaly detection, due to their high power of representation and and generative ability for tasks like root cause analysis and simulation.

Having said that, it's evident from the comparison and bennchmarking performed with the SVM model that GAN methods come at the high cost of added complexity, which makes model design and fine tuning considerably more difficult, initially degrading performance. For this reason, we recommend considering a GAN based approach in the right circumstances, namely when the available data is appropriate and when the expected performance gain justifies the added complexity.

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio (2014) *Generative Adversarial Networks*, arXiv:1406.2661.

[2] Martin Arjovsky, Soumith Chintala, Léon Bottou (2017) *Wasserstein GAN*, arXiv:1701.07875.

[3] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, Georg Langs (2017) *Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery*, arXiv:1703.05921.

[4] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville (2017) *Improved Training of Wasserstein GANs*, arXiv:1704.00028.

[5] Yunus Saatchi, Andrew Gordon Wilson (2017) *Bayesian GAN*, arXiv:1705.09558.

[6] Tero Karras, Timo Aila, Samuli Laine, Jaakko Lehtinen (2017) *Progressive Growing of GANs for Improved Quality, Stability, and Variation*, arXiv:1710.10196.

[7] Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, Vijay Ramaseshan Chandrasekhar (2018) *Efficient GAN-Based Anomaly Detection*, arXiv:1802.06222.

[8] Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, Liqiang Wang (2018) *Improving the Improved Training of Wasserstein GANs: A Consistency Term and Its Dual Effect*, arXiv:1803.01541.

[9] Dan Li, Dacheng Chen, Jonathan Goh, See-kiong Ng (2018) *Anomaly Detection with Generative Adversarial Networks for Multivariate Time Series*, arXiv:1809.04758.