Latent Space Anomaly Detection with Bidirectional GANs

Thijs van Ede
t.s.vanede@utwente.nl
University of Twente

February 11, 2019

Abstract

This work proposes two methods for the use of Bidirectional Generative Adversarial Networks for anomaly detection (AD-GANs). The first method uses the encoding and generating properties to reconstruct original data. If this reconstruction fails, the data is concidered anomalous. The second method applies traditional anomaly detection techniques in the latent space generated by the encoder of a Bidirectional GAN. Both methods outperform traditional anomaly detection in low dimensional space but require additional analyses before being applicable in practice.

1 Introduction

The demand for well performing anomaly detection techniques comes from - amongst others - computer security research, specifically from the field of intrusion detection. Present day internet usage consists of network traffic generated by a vast amount of different devices, applications and usage patterns. From a security perspective, such chaotic datastreams are undesirable as it becomes difficult to identify the different sources of this data and their intentions. Specifically, it becomes increasingly harder to detect malicious network communication and block it before it is able to perform all its malicious communication.

In order to detect malicious communication, the security community created Network Intrusion Detection Systems (NIDSs). Traditionally, these systems were fed signatures for malicious programs based on distinct patterns in the programs behaviour [12, 14, 15]. E.g. common strings or statistical patterns found in the network packets. However, as increasingly more malware is generated, this is a tedious and inefficient method of detection due to the required amount of manual malware analysis. Therefore, more recent research focuses on building NIDS based on anomaly detection in network traffic [1, 4]. In this scenario, a model of the normal behaviour is constructed and any deviating behaviour is concidered malicious. While this method is more resilient against new malware, it requires expert knowledge on the construction of a normal behaviour model as raw network traffic contains too many dimensions as well as obsolete information for anomaly detection.

To this end we propose two variants of Bidirectional Generative Adversarial Networks (BiGANs) [2] to perform anomaly detection. To show the effectiveness of anomaly detection in general, we evaluate these variants of the BiGANs on the MNIST dataset of handwritten images. The two proposed Anomaly Detection GANs (AD-GANs) include leveraging the encoding and reconstruction capabilities (Section 3.1) and the application of traditional anomaly detection techniques within the latent space (Section 3.2). The idea here is that the latent space contains a low-dimensional representation of the original data, where previously unseen classes will be mapped to new areas of the latent space.

The contribution of this work therefore lies in the adaptation of Bidirectional GANs for the purpose of anomaly detection. Specifically, the architecture and evaluation of the two proposed AD-GANs.

2 Related Work

After the invention of Generative Aversarial Networks (GANs) [5], many variants have been introduced that focus on specific applications. In this Section, we will give a brief overview of the most interesting variants to the application to NIDS in Section 2.1 and elaborate on the use of BiGANs [2] in Section 2.2.

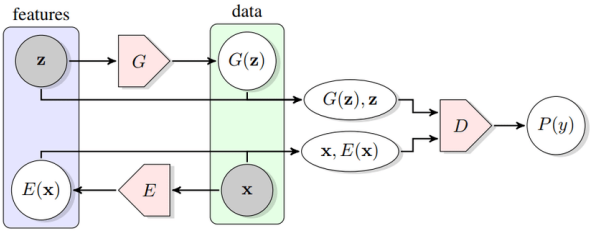


Figure 1: Overview of BiGAN architecture.

2.1 GANs for Anomaly Detection

The traditional GAN proposed by Goodfellow et al. [5] alternates between training a Generator network $\mathcal{G}$ and discriminator network $\mathcal{D}$. The task of $\mathcal{G}$ is to create realistic data from randomly generated noise, while the task of $\mathcal{D}$ is to discriminate between real data and fake data generated by $\mathcal{G}$. In the ideal case the generator $\mathcal{G}$ creates increasingly more realistic data and the discriminator $\mathcal{D}$ will have increasingly more difficulty discriminating between real and generated data. While this system is able to create realistic data, it does not have the ability to create data of novel classes. This is due to the fact that there are no constraints put on the latent (noise) representation. Furthermore, it is only capable of generating realistic data from noise and cannot create a latent representation of actual data.

To tackle this problem, several other methods have been proposed. In order to transform real data to a latent representation Adversarially Learned Inference (ALI) models [3] and BiGANs [2] have been introduced. Both approaches introduce an encoder $\mathcal{E}$ that learns a latent representation of the actual data. In addition, the discriminator $\mathcal{D}$ bases its decision on both the latent representation and the actual data to obtain an optimal bidirectional mapping from data to the latent representation. We will elaborate more on BiGANs in Section 2.2

Next to learning a mapping to the latent representation, an AD-GAN must also be able to discriminate classes, e.g. in the latent representation. Ideally, the mapping from real data to the latent dimension should map novel classes to previously unused regions of the latent space. Conditional GANs [9] were introduced, which modify the generator $\mathcal{G}$ to take both random noise and a class label as input. This can be seen as adding an additional dimension to the latent space which includes the class label. A similar technique was used in Auxiliary Classifier GAN [11]. Furthermore, other techniques that attempt to generate meaningful latent representations have been developed based on variational autoencoders [8]. Unfortunately, none of the approaches discuss the possibility of adding novel classes, nor are able to map existing data to the latent representation.

2.2 BiGAN

Donahue et al. [2] introduced the Bidirectional GAN (BiGAN). As previously stated, the general idea is that an encoder network is added which learns a latent representation of the real data. Subsequently, the discriminator $\mathcal{D}$ requires as input both the latent representation and the data. That is, for the generated data it receives the input $(\mathcal{G}(\mathbf{z}), \mathbf{z})$ and for actual data it receives the input $(\mathbf{x}, \mathcal{E}(\mathbf{x}))$. Finally, the discriminator $\mathcal{D}$ tries to output 1 if the data and latent representation are real and 0 if the data and latent representation originate from the generator $\mathcal{G}$. Figure 1 gives an overview of the BiGAN architecture.

The original BiGAN article discusses the use of latent representation for classification. In order to classify new datasamples, they encode a latent representation of labelled samples and perform One Nearest Neighbors (1NN) classification which achieves a 97.39% accuracy. This performance is competitive with other techniques for creating a meaningful latent space, i.e. similar classes are clustered together. Hence, it is a promising technique in which to apply clustering techniques, used in many anomaly detec-

tion systems.

Unfortunately, Donahue et al. do not discuss the application of BiGANs for anomaly detection. A reason for this might be that the encoder and generator learned by the BiGAN use the full latent space for the known classes leaving no unused regions that novel classes may occupy. In addition, other works [10] suggested that the latent representation might not be meaningful as written in the original paper. To verify this observation we scrutinise the latent space generated by the MNIST dataset and the mapping to this space by novel classes in Section 4.

3 Anomaly Detection GAN

In order to determine the extent to which BiGANs can be used for anomaly detection in network traffic we propose two variants of the Anomaly Detection GAN (AD-GAN). Our first approach is based on a regular BiGAN and its encoding and generating properties and will be discussed in Section 3.1. The second approach - discussed in Section 3.2 - attempts to construct a latent space suitable for applying known anomaly detection methods. We evaluate the effectiveness of our AD-GANs using the MNIST dataset [6] in Section 4.

3.1 Reconstruction-Based AD-GAN

The general idea of a Reconstruction-Based AD-GAN[1] is that for data point x, the subsequent application of encoder and generator $\mathcal{G}(\mathcal{E}(x))$ yields data similar to the input if the class of x was in the training data. Conversely, if the class of x was *not* in the training data, the reconstruction will be poor as both encoder and generator were not trained with this data. To this end, the Reconstruction-Based AD-GAN is trained in the same way as a regular BiGAN. In the anomaly detection phase, the system evaluates the mean squared error loss between x and $\mathcal{G}(\mathcal{E}(x))$. If this loss function exceeds a given threshold, the data is concidered anomalous, otherwise it is labelled as normal.

[1] An implementation can be found at `https://github.com/Thijsvanede/GANs`

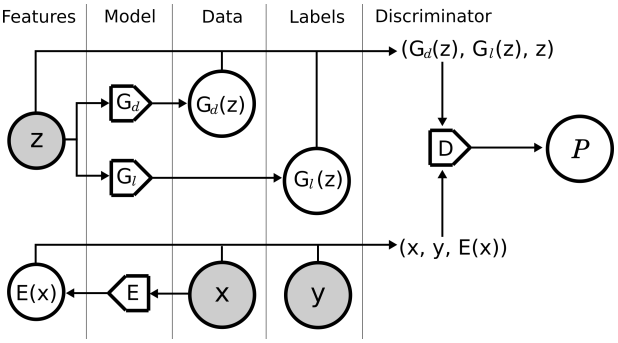


Figure 2: Overview of Class-Based AD-GAN architecture. The grey nodes indicate the input into the system.

3.2 Class-Based AD-GAN

A second approach is to encorporate class labels into a regular BiGAN. In this scenario, the discriminator $\mathcal{D}$ bases its decision not only on the 2-tuple (data, latent), but on the 3-tuple (data, label, latent). Figure 2 shows an overview of a Class-Based AD-GAN[1]. Here, the encoder supplies the discriminator with the tuple $(x, y, \mathcal{E}(x))$, where x is the actual data, y, the actual label, and $E(x)$ the encoded latent representation of the data. The generator on the other hand is split up into two separate generators, $\mathcal{G}_d$ generating data from the random noise input and $\mathcal{G}_l$ generating a label from the same random noise input. This generator part of the GAN inputs the tuple $(\mathcal{G}_d(z), \mathcal{G}_l(z), z)$ into the discriminator.

The rationale behind this architecture is that if the labels have to be created from the latent space along with the data, the generator $\mathcal{G}_d$ is forced to generate different classes from different regions in the latent space. This in turn should lead to a better separable latent space in terms of classes which makes it possible to apply traditional anomaly detection techniques on the latent space.

3.2.1 Related Work

We note that the Class-Based AD-GAN has similarities with other types of GAN. This Section attempts to clarify the differences between existing im-

plementations and our newly introduced Class-Based AD-GAN architecture. First, the Auxiliary Classifier GAN (AC-GAN) [11] and Conditional GAN (CGAN) [9] add an additional input to its generator $\mathcal{G}$ forcing it to produce output corresponding to the given class label. Similar to our approach, the discriminator takes as input both the data and class. Conversely, the approaches differ from ours as they basically add dimensions to its latent space which represent the class. It does not force the generator to separate the latent space itself. Another similar approach is found in the Semi-supervised GAN [10] which encorporates class labels by letting the discriminator predict the class label instead of a real versus fake prediction. All of the previously named approaches also differ in that they do not have an encoder, nor do they add a latent space to the discriminator. However, the addition of an encoder is a trivial extension when combining the approaches with a BiGAN.

4 Evaluation

In order to evaluate both the Reconstruction-Based AD-GAN and Class-Based AD-GAN, we compare them with a baseline set by PCA based anomaly detection. This is all done on the same MNIST dataset of handwritten digits.

4.1 Dataset

The MNIST dataset [6] consists of 60,000 labelled 28x28 images of handwritten digits for training and 10,000 labelled images for testing. This data is subdivided into 10 classes of digits 0 through 9. Anomaly detection requires that there are new classes of data in the testing phase compared to the training phase. These new classes are concidered anomalies. We define the following terminology: classes and their corresponding data items that have been seen during training are reffered to as *known* classes, whereas all classes not seen during training are referred to as *unknown* classes.

In order to evaluate the performance of all anomaly detectors, we randomly select 2 classes in the training part of the dataset to be left out. This results in all detectors being trained by 8 *known* classes. During the testing phase, all 10 classes will be evaluated, i.e. 2 *unknown* classes and 8 *known* classes. Note that the same choice of *known* and *unknown* classes are used across all experiments.

4.2 Experimental Setup

For our experiment, we attempt to create equivalent setups for all three anomaly detection systems. We have chosen to work with a latent space with dimension 2. We have chosen such a low number to demonstrate the effectiveness even in low-dimensional space. Furthermore, a 2 dimensional space can be better represented visually, giving more insights in the distribution within the latent representation.

4.2.1 PCA Anomaly Detection

As this experiment works with a latent space of dimension 2, we use PCA to reduce the input images to its 2 most important principal components. Next, we apply an out of the box isolation forest [7] in order to perform anomaly detection. As in all experimental MNIST setups, the detector is trained using the 8 *known* training classes and tested using all 10 testing classes.

4.2.2 Reconstruction-Based AD-GAN

For the Reconstruction-Based AD-GAN, we setup the parts of the network as described in Table 1. The network was trained to minimise the binary crossentropy loss of the discriminator as it should decide only between 0 for fake or 1 for real data. The Adam optimiser was used with a learning rate of 0.001, first order momentum of 0.5 and second order momentum of 0.999. These values were modeled on an existing BiGAN implementation[2]. Finally, the anomaly detector was trained using the *known* MNIST training data for 10,000 iterations of batch size 64.

As previously stated, the anomaly detection of our Reconstruction-Based AD-GAN is based on the

[2]https://github.com/eriklindernoren/Keras-GAN

Layer	Type	Dimensions	Activation
Generator $\mathcal{G}$			
Input	Input	2	
Hidden	Dense Network	512	Leaky ReLU
Hidden	Batch Normalisation		
Hidden	Dense Network	512	Leaky ReLU
Hidden	Batch Normalisation		
Output	Dense Network	28x28	Tanh
Encoder $\mathcal{E}$			
Input	Input	28x28	
Hidden	Dense Network	512	Leaky ReLU
Hidden	Batch Normalisation		
Hidden	Dense Network	512	Leaky ReLU
Hidden	Batch Normalisation		
Output	Dense Network	2	
Discriminator $\mathcal{D}$			
Input	Input	(28x28, 2)	
Hidden	Dense Network	1024	Leaky ReLU
Hidden	Dropout		
Hidden	Dense Network	1024	Leaky ReLU
Hidden	Dropout		
Hidden	Dense Network	1024	Leaky ReLU
Hidden	Dropout		
Output	Dense Network	1	Sigmoid

Table 1: Overview of Reconstruction-Based AD-GAN architecture.

reconstruction performance of encoder and generator. We measure the reconstruction performance as the mean squared error between datapoint x and $\mathcal{G}(\mathcal{E}(x))$. If this value is above a predefined threshold τ we label the sample as anomalous. Otherwise it is classified as benign. For our experiment we used values of $\tau = 0.5, \tau = 0.8$ and $\tau = 0.9$. The results are displayed in Section 4.4.

4.3 Class-Based AD-GAN

The Class-Based AD-GAN has a similar architecture to that of the Reconstruction-Based AD-GAN of Table 1. However its label generator $\mathcal{G}_l$ has an output dimension of 8, i.e. one for each training class and softmax activation. Furthermore, the discriminator $\mathcal{D}$ accepts an input of (28x28, 8, 2) for it requires the class label as a second input to the tuple. All other parameters are equal to that of the Reconstruction-Based AD-GAN, including the training iterations, optimiser and loss function.

Detector	TP[3]	TN[4]	FP[5]	FN[6]	F1	Acc
PCA	6301	214	1710	1775	78.34%	65.15%
$RB_{0.5}$	3953	1348	576	4123	62.72%	53.01%
$RB_{0.8}$	7566	91	1833	520	86.53%	76.47%
$RB_{0.9}$	7928	14	1910	148	88.51%	79.42%
CB	6609	305	1619	1467	81.07%	69.14%

Table 2: Comparison between PCA based (PCA), Reconstruction-Based AD-GAN (RB) and Class-Based AD-GAN (CB) anomaly detectors. Showing number of True/False Positive/Negative samples, F1 score and accuracy.

In order to perform anomaly detection, we encode all testing samples to the latent space and apply the same anomaly detection technique as used in the PCA Anomaly Detection of Section 4.2.1, namely the isolation forest.

4.4 Results

Table 2 displays the results of all three anomaly detectors. In this table we can see that regular PCA anomaly detection performs worse than most Reconstruction-Based AD-GANs and the Class Based AD-GAN. Unfortunately, all results are insufficient for realistic application as all classifiers have difficulty detecting novel samples and simultaniously include high false negative rates. This low performance may be attributed to the reduction to a 2 dimensional space.

5 Discussion

The performance of both proposed approaches is far from perfect. In this section we discuss some possible reasons of the poor results and suggest future work for improving the proposed systems.

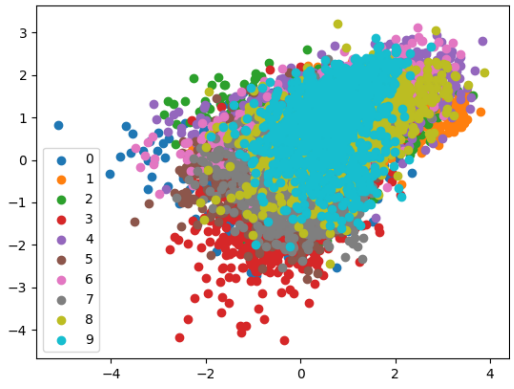
[3]True positives, known classified as known.
[4]True negatives, unknown classified as unknown.
[5]False positives, unknown classified as known.
[6]False negatives, known classified as unknown.

5.1 Higher Dimensionality

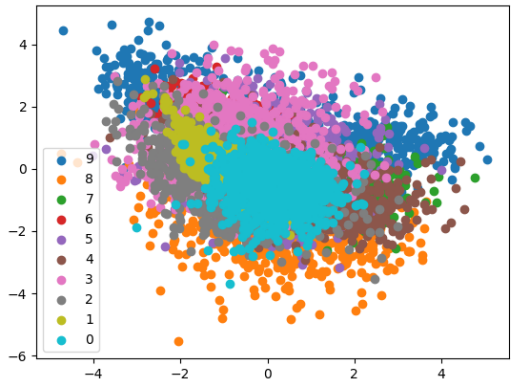
The analysis performed in this research was applied to a highly reduced feature space, i.e. 784 dimensions were compressed to 2 dimensions. By compressing the data into a higher dimensionality than two, we are likely to observe better results, both for the traditional PCA based anomaly detection as well as the AD-GANs. Hence, future work should investigate the performance of AD-GANs with a higher dimensional latent space.

5.2 Reconstruction-Based AD-GAN

The Reconstruction-Based AD-GAN was based on the assumption that a BiGAN overfits on the data it is trained with. Given this proposition, the encoder and generator of the BiGAN should not be able to reconstruct novel classes. However, the results in Table 2 show that reconstruction of *unknown* samples is still possible. Suppose we take $\tau = 0.8$, 91 out of 1924, i.e. 4.73% of *unknown* samples could not be reconstructed to the desired threshold versus 520 out of 8086 = 6.43% of *known* samples. For the other versions, results show an even greater imbalance. From this result, we conclude that a BiGAN generalises too well for a Reconstruction-Based AD-GAN to be infeasible. In order to improve the performance of a Reconstruction-Based AD-GAN, one could employ fewer regularisation techniques than used in our experiments. E.g. remove Batch Normalisation or use different optimisation functions that discard any form of weight decay.



(a) BiGAN



(b) Class-Based AD-GAN

Figure 3: Latent representation of test data.

5.3 Class-Based AD-GAN

The Class-Based AD-GAN performs anomaly detection on the latent representation of data. For this to work, the latent representation of data should be well-separated per class. Figure 3 shows a comparison of the latent space of a regular BiGAN and a Class-Based AD-GAN. From this Figure, we can see that the latent space of the Class-Based AD-GAN is slightly better separated than that of the BiGAN. However, in order to perform anomaly detection, the classes must be separated even further such that novel classes can take up unoccupied regions.

5.3.1 Label Distribution

The poor separation of the latent space could be due to a number of factors. First, the Class-Based AD-GAN is not incentivised to generate a representable distribution of classes. To illustrate this, imagine a generator pair $\mathcal{G}_d$, $\mathcal{G}_l$ that only outputs data that looks like a 0 with that same class. The discriminator will have difficulty distinguishing this from actual

data without the generator, and in extend the encoder being able to generate any other numbers. By analysing the Class-Based AD-GAN we found that the most likely label it generated was a 4 in 37.6% of cases versus the least likely label 6 in only 0.3% of cases even though the distribution of all numbers was equal during training. Therefore, future work should look into incentivising the generation of a more uniform class distribution by $\mathcal{G}_l$. We propose that this can be done through occasionally replacing the tuple $(\mathcal{G}_d(z), \mathcal{G}_l(z), z)$ with $(\mathcal{G}_d(\mathcal{E}(x)), y, \mathcal{E}(x))$. Another approach may be used by training with a combined loss function of both the discriminator and a function for the desired prior distribution on the latent space alike Adversarial Autoencoders [8].

5.3.2 Other Use Cases

Despite the Class-Based AD-GAN being far from perfect, it has other interesting use cases apart from anomaly detection. If the latent space can be well-separated, it is possible for the algorithm to produce new data of certain classes like the Conditional GANs [9]. Forthermore a system with a well-separated latent space can be used for e.g. image synthesis [13] and image-to-image translation [16].

6 Conclusion

While both Reconstruction-Based and Class-Based AD-GANs outperform traditional anomaly detection systems, both have several limitations that need to be overcome. Reconstruction-Based AD-GANs seem to generalise their encoding and generation of data too well and therefore need to be adapted in order to detect anomalies. Class-Based AD-GANs on the other hand seem to require more incentives in order to generate a well separated latent dimension. Hence, using BiGANs for anomaly detection requires more research before it can be used in practice.

References

[1] Riccardo Bortolameotti, Thijs van Ede, Marco Caselli, Maarten H Everts, Pieter Hartel, Rick Hofstede, Willem Jonker, and Andreas Peter. Decanter: Detection of anomalous outbound http traffic by passive application fingerprinting. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 373–386. ACM, 2017.

[2] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[3] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[4] Pedro Garcia-Teodoro, Jesus Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1-2):18–28, 2009.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[6] Ernst Kussul and Tatiana Baidyk. Improved method of handwritten digit recognition tested on mnist database. *Image and Vision Computing*, 22(12):971–981, 2004.

[7] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[8] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[9] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[10] Augustus Odena. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

[11] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2642–2651. JMLR. org, 2017.

[12] Roberto Perdisci, Wenke Lee, and Nick Feamster. Behavioral clustering of http-based malware and signature generation using malicious network traces. In *NSDI*, volume 10, page 14, 2010.

[13] Scott E Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning what and where to draw. In *Advances in Neural Information Processing Systems*, pages 217–225, 2016.

[14] Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In *Lisa*, volume 99, pages 229–238, 1999.

[15] Vincent F Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 439–454. IEEE, 2016.

[16] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*, 2017.