Anomaly Detection using GANs

DHANUNJAYA ELLURI THIMMARAJU, Technische Universität Dortmund, Germany

Anomaly detection is the most explored area in various domains including manufacturing, cybersecurity, computer vision, etc. Although the supervised models are able to produce promising results, the most significant challenge is detecting the unseen anomalies during test time. In this paper, we survey unsupervised techniques for detecting anomalies that use Generative Adversarial Networks (GANs) based architectures highlighting its pros and cons.

Additional Key Words and Phrases: Anomaly Detection, Generative Adversarial Networks, Unsupervised Learning, Autoencoders

1 INTRODUCTION

Anomalies are abnormal patterns in the data that do not confirm the standard notions of normal behavior in the data [3]. Despite yielding encouraging results, the supervised models require large labeled and annotated datasets and fail to identify the unseen anomalies. Generative Adversarial Networks (GANs) [7] are a class of models that are successfully able to model the complex and high dimensional distributions of data. Though the GANs are widely used to generate synthetic data, the application of GANs for anomaly detection is recently explored.

Intuitively, the GANs trained to fit the distribution of normal samples should be able to reconstruct the normal sample from some latent distribution (noise prior) and should be able to discriminate between the samples from the true and latent distribution. According to our understanding, the detection of anomalies using GANs is built upon the Adversarial Feature Learning idea in which the GANs are trained to inverse map from generated samples to latent samples. Alternatively, the BiGANs are proposed for this task. A learned model that is able to map from input data to latent representation which does the opposite of generator is the idea behind the anomaly detection using GANs.

Apart from the introduction section, this paper is divided into further 4 sections. In Section 2, we introduce the architecture and working principle of GANs and its extension namely BiGANs. In Section 3, we introduce the different architectures of GANs proposed for anomaly detection. In Section 4, we empirically assess the performances of different proposed models. Finally, in Section 5, we draw the conclusions and discuss further research recommendations.

2 INTRODUCTION TO GANS

2.1 GANs

Generative Adversarial Networks (GANs) [7] consist of two models, the Generator G and the Discriminator D that are trained against each other simultaneously in an adversarial manner. The purpose of generator G is to fool the discriminator by generalizing the true distribution to generate some random images similar to true images given some latent distribution samples, and the discriminator D is to differentiate between the samples coming from latent distribution and generated distribution. To generalize the true distribution x, the generative model learns to construct a generative distribution p_g by mapping from latent distribution (noise prior) p_z to the data space $G(z, \theta_g)$ where θ_g are the generator parameters. The discriminator produces a probabilistic value representing the value that the input to this model comes from true distribution rather than the generative distribution. The discriminator is denoted as $D(x, \theta_d)$ where the θ_d are the discriminator parameters. The architecture of the GANs can be depicted in the following Figure 1.

We train the discriminator to assign the correct probabilities to differentiate between the true samples and generated samples by maximizing the $D(x)$. Simultaneously, the generator is trained to minimize $log(1 - D(G(z)))$ to generate samples $G(z)$ similar to

Author's address: Dhanunjaya Elluri Thimmaraju, Technische Universität Dortmund, Dortmund, Germany, dhanunjaya.elluri@tu-dortmund.de.

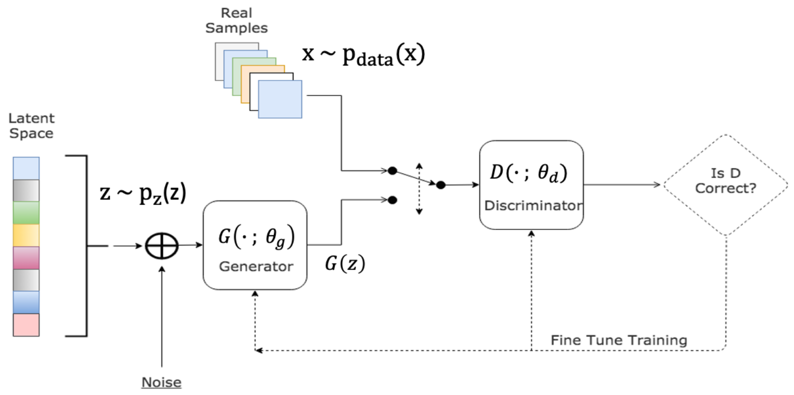

Fig. 1. Generative Adversarial Networks [6]

the true distribution p_x. The authors of the GANs posed this problem as a two-player minimax game where the two players D and G compete against each other represented by a value function $V(D,G)$.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))] \tag{1}$$

The optimization of GANs is done by the gradient descent approach. However, optimizing both discriminator D and the generator G simultaneously would lead to overfitting and the discriminator would reach its maximum optimal state where the generator could never fool D. Instead, we limit the optimization of D for some k steps and one step of optimization of G over some m steps of training.

2.2 Conditional GANs

The working principle of GANs can be extended by introducing some extra information y as a condition in either generator G or the discriminator D [9]. They can be any auxiliary information such as labels or data from other modalities. This condition can be incorporated into the model as an extra layer in G or D. The generator combines the noise prior $p_z(z)$, and the conditioned information y into a joint hidden representation and the adversarial learning process emphasizes how this hidden representation is composed. The overall objective of these Conditional GANs can be represented as the following function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x|y)}[log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z|y)))] \tag{2}$$

2.3 BiGANs

Whilst the Generative Adversarial Networks (GANs) are trained to learn the true distribution by mapping from latent space to true data space, Bidirectional GANs (BiGANs) [4] add one more model namely Encoder to the GANs to enable the model to do inverse mapping (opposite of generator $E = G^{-1}$) from generated space to latent space. This enables the BiGANs to learn the rich feature representations given some arbitrary distribution. The overall architecture for the BiGANs is depicted in Figure 2.

The way the Encoder E is incorporated into the GANs is by changing the discriminator function. The BiGAN discriminator D discriminates not only in the data space (x versus $G(z)$) but the joint distribution of data and encoded space (tuples x and $E(x)$ versus $G(z)$ and z) where the latent component is either from an encoder $E(x)$ or the generator input z.

$$\min_{G,E} \max_D V(D,G,E) = \mathbb{E}_{x\sim p_{data}(x)}[log(D(x,E(x)))] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z),z))] \tag{3}$$

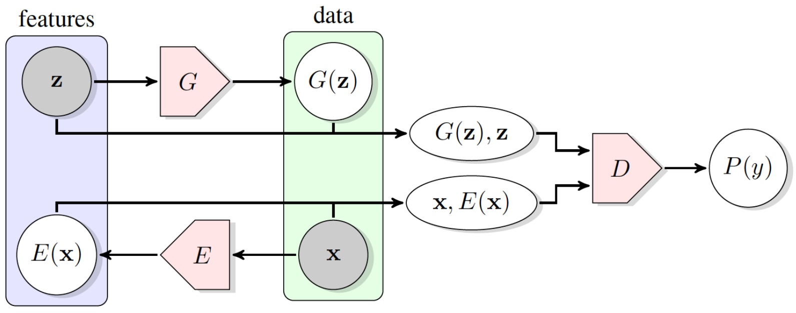

Fig. 2. The structure of Bidirectional GANs [4]

3 GANS FOR ANOMALY DETECTION

Anomaly detection using GANs was first proposed by the authors [11] in their paper referred to as AnoGAN. In order to face the performance issues with AnoGAN, a BiGAN based architecture is introduced in the paper referred to as EGBAD (Efficient GAN Based Anomaly Detection) [12]. This architecture outperformed the AnoGAN both in accuracy and performance. In recent years, a new advanced GAN + Autoencoder [2] architecture is proposed, which outperforms EGBAD in performance. In the following sections, we present the analysis of each architecture mentioned and discuss the pros and cons.

3.1 AnoGAN

AnoGANs utilize the Deep Convolutional GANs (DCGANs) [10][11] which learn to map from latent space z to the generated sample space $\hat{x} = G(z)$ and use this learned model to do the inverse mapping from given new, unseen samples back to the latent representation. To identify anomalies, AnoGANs are trained only on positive samples (non-anomalous samples), which makes the generator learn the manifold χ of generated positive samples. Given the model to generate normal samples, when an anomalous image is fed to the generator, the reconstruction image would be non-anomalous. This helps us to identify and localize the anomalies in images. This two-step training and detection of anomalies are depicted in Figure 3.

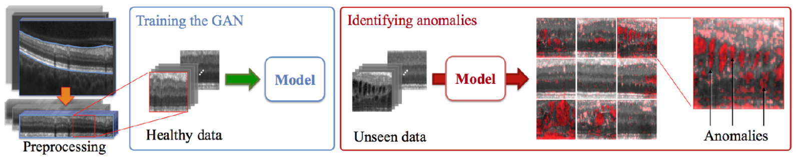

Fig. 3. The two-step process of anomaly detection [11]

GANs are trained only to map from latent space to realistic space $G(z) = z \longrightarrow x$ and by default it does not support inverse mapping $u(x) = x \longrightarrow z$. To do this inverse mapping given a query image x, we find a random point z in the latent space which corresponds to the generation of image $G(z)$ which is visually similar to the query image. However, since the latent space follows a smooth transition, sampling from two points close to each other would generate visually similar images. Hence, to find the best z in the latent space, we start with random point z_1 in the latent space and feed it to the generator to get the generated image $G(z_1)$. We define a new loss function to measure the similarity and provide the gradients to update the z_1 to z_2 in latent space. In order to find the best z_Γ, we iteratively search the best point in latent space in some iterative steps $\gamma = 1, 2, \ldots, \Gamma$.

For mapping of new, unseen images back to the latent space, we define a new loss function that comprises of two components, *residual loss*, and *discriminator loss.*

Residual Loss: The residual loss is used to measure the visual similarity between the generated image $G(z_\gamma)$ and the query image x and is defined as follow:

$$\mathcal{L}_R(z_\gamma) = \|x - G(z_\gamma)\|_1 \tag{4}$$

Discriminator loss: In contrast to the discriminator model, where the z_γ is updated to fool the discriminator, we define an alternative discriminator loss to match the $G(z_\gamma)$ to the query image x. Here, we propose a loss function for richer intermediate feature representation of the discriminator, which is defined as follows:

$$\mathcal{L}_D(z_\gamma) = \|f(x) - f(G(z_\gamma))\|_1 \tag{5}$$

where, $f(.)$ represents the intermediate layer of the discriminator.

Hence, for mapping back to the latent space, the overall loss is defined as the weighted sum of the above two loss functions.

$$\mathcal{L}(z_\gamma) = (1-\lambda) \cdot \mathcal{L}_R(z_\gamma) + \lambda \cdot \mathcal{L}_D(z_\gamma) \tag{6}$$

The anomaly score, which defines the fit of a query image to the normal sample is directly inherited from the loss function. Thus, the anomaly score has no upper bound and a higher anomaly score corresponds to a higher probability that the query image x is anomalous.

$$A(x) = \mathcal{L}(z_\Gamma) \tag{7}$$

Pros:

1. It is the first proposed model that showed GANs can be used for anomaly detection.
2. Introduced a new mapping schema from latent space to real space. Defined an anomaly score from the same loss function.

Cons:

1. Requires Γ iteration steps to backpropagate to find the best z_γ in latent space.
2. Since the anomaly score has no upper bound, it is difficult to interpret the results.

3.2 EGBAD

In order to deal with the performance issues of AnoGAN, the Efficient GAN Based Anomaly Detection (EGBAD) [12] introduces the BiGAN architecture to do the inverse mapping. EGBAD also utilizes the same concept of AnoGAN where the model is trained only on normal samples to generate a manifold χ of generated normal samples and detects the anomalies by passing the anomalous images to the generator which in turn generates the non-anomalous images that help us to identify the anomalies. Unlike GANs, the BiGANs incorporate the Encoder E that maps input samples x to the latent space z, which helps to avoid the computationally expensive step of recovering the feature representation at test time. In this context, the discriminator D not only considers input samples (real or fake) but a joint distribution of input samples and feature representation (either the encoded representation or generator input). The optimization steps for min-max $V(D,G,E)$ is adopted from the similar work of the authors [4][5], where the value function $V(D,G,E)$ is defined as in equation 3. The major contribution of EGBAD is to eliminate Gamma backpropagation steps involved in AnoGANs and compute anomaly scores without these Gamma steps.

3.3 GANomaly

Inspired by the works on AnoGAN and EGBAD, the new authors proposed a semi-supervised conditional GANs called GANomaly architecture for anomaly detection [2]. This architecture was introduced to address the instability of GANs training, which involves in two-stage training for mapping of latent space to real and vice versa. Unlike GANs where the generators are modeled as decoders, GANomaly introduces autoencoders architecture into the generators. The overall architecture of GANomaly is depicted in the following Figure 4 and is explained as below:

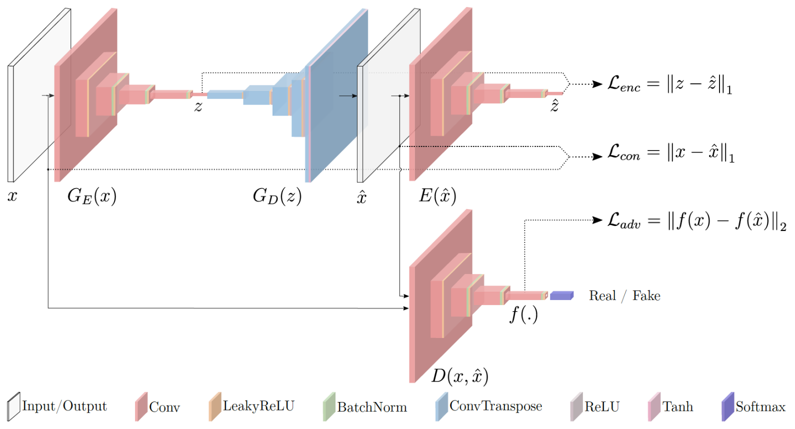

Fig. 4. GANomaly architecture [2]

Generator Network: The generator network is comprised of Autoencoders + Decoder as shown in Figure 3. In the autoencoder part, the encoder G_E learns to map the input samples x to latent space z whereas the decoder G_D learns to reconstruct the latent representation z back to its real space $\hat{x}$. On the other hand, the second encoder, which has the same architecture of G_E learns to map generated real samples $\hat{x}$ to the encoded representation $\hat{z}$. The main contribution of this generator network is to learn not only better reconstruction of images by the utilization of autoencoder but also to establish the control on latent space by incorporating the second encoder into the generator network. Given that the model is learned to reconstruct the images from x to $\hat{x}$ if the input image is anomalous, the reconstructed image $\hat{x}$ would be different from x, and in particular, helps to localize the anomalies.

Discriminator Network: The discriminator network is directly inherited from the original GANs where this network is used to discern between the real samples and fake samples.

In order to learn the better reconstruction of images and control over the latent space generation, GANomaly introduced the *generator loss* function as the sum of the three loss functions, each of which helps to optimize the individual sub-networks.

Adversarial loss: The adversarial loss is introduced to learn feature matching to reduce the instability of GAN training. Unlike GANs, where the generator is updated based on the output of the discriminator, here the G is updated based on the internal representation of D.

$$\mathcal{L}_{adv} = \mathbb{E}_{x \sim px} \| f(x) - \mathbb{E}_{x \sim px} f(G(x)) \|_2, \tag{8}$$

where $f(.)$ is a function that outputs the intermediate layer of the discriminator.

Contextual loss: The contextual loss learns the contextual information between the real and generated samples and ensures the visual similarity between these images.

$$\mathcal{L}_{con} = \mathbb{E}_{x \sim px} \| x - G(x) \|_1. \tag{9}$$

Encoder loss: The two loss functions introduces above ensures not only the visual similarity between generated and real samples but also contextually sound. The encoder loss here ensures how to best encode the normal images.

$$\mathcal{L}_{enc} = \mathbb{E}_{x \sim px} \| G_E(x) - E(G(x)) \|_2. \tag{10}$$

The overall generator loss function is the weighted sum of the above three loss functions.

$$\mathcal{L} = \omega_{adv} \mathcal{L}_{adv} + \omega_{con} \mathcal{L}_{con} + \omega_{enc} \mathcal{L}_{enc}, \tag{11}$$

where $\omega_{adv}, \omega_{con}, \omega_{enc}$ are the learned parameters to adjust the importance of three losses.

Anomaly score: Unlike the anomaly score defined in AnoGAN, which considers both residual and discriminator losses into consideration, here the anomaly score is calculated using only the encoder loss.

$$\mathcal{A}(x) = \|G_E(x) - E(G(x))\|_2. \tag{12}$$

In order to better interpret the anomaly score, the score is computed to every test sample $\hat{x}$ within the test set $\hat{\mathcal{D}}$ which yields a set of anomaly scores $\mathcal{S} = \{s_i : \mathcal{A}(\hat{x}_i), \hat{x}_i \in \hat{\mathcal{D}}\}$ and apply a feature scaling on this set to make the score fall in probabilistic values of range $[0, 1]$.

$$s_i' = \frac{s_i + min(\mathcal{S})}{max(\mathcal{S}) - min(\mathcal{S})}. \tag{13}$$

Pros:

1. Usage of autoencoders makes the entire learning process of the model faster.
2. The anomaly score is easier to interpret.
3. No backpropagation is required as in AnoGANs.

Cons:

1. Defines a new anomaly score.
2. Could not reach the accuracy levels of EGBAD.

4 EXPERIMENTS AND RESULTS

To distinguish between the performances of each model presented in this paper, the authors of the paper A Survey on GANs for Anomaly Detection [8] published the results on datasets MNIST and FMNIST which consists of 10 labels in each of them in their paper. All the models were reimplemented using the deep learning framework Tensorflow [1] and is available at the following GitHub repository.
https://github.com/zurutech/anomaly-toolbox

In order to test for a broader range of model configurations, the GANs are trained with different hyperparameters. All the test results were measured under AUPRC (area under precision and recall curve). The obtained results are illustrated in the Figure 5. It is observed that the BiGANs/EGBAD outperformed GANomaly.

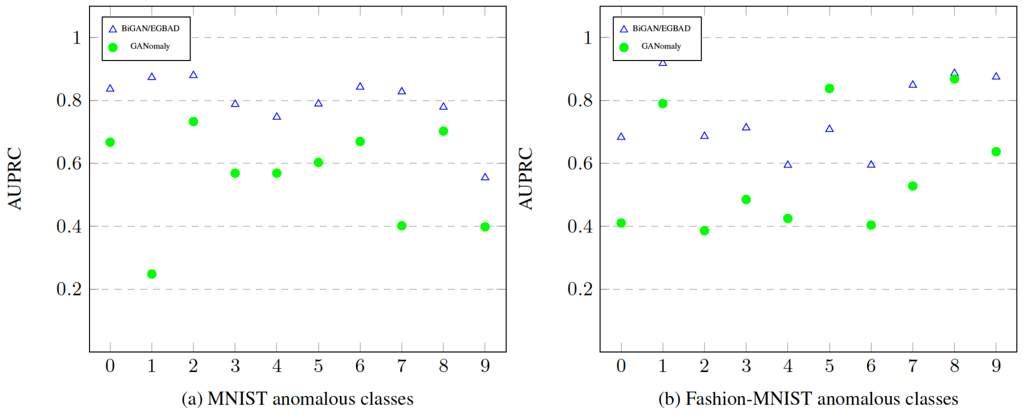

Fig. 5. Comparison of performances of EGBAD and GANomaly on MNIST and FMNIST datasets [8]

5 SUMMARY

This report focuses on understanding the concepts of Generative Adversarial Networks (GANs) and its extended versions of Conditional GANs and BiGANs and their applications for anomaly detection in images. Although the GANs are developed for generating synthetic data, their application to anomaly detection is recently explored.

In this paper, we examined the first proposed model AnoGANs for anomaly detection and discussed the drawbacks of Γ iteration steps to find the best z in latent space that corresponds to the generation of an image similar to the given query image. To overcome this problem, a BiGAN based EGBAD model is proposed for inverse mapping without the Γ iteration steps. Although it is efficiently able to learn the inverse mapping, both the architectures follow the two-step process for anomaly detection. Then, an advanced autoencoder + decoder architecture GANomaly is proposed to eliminate this instability of training by learning of better reconstruction of images and control over the feature learning.

Overall, we demonstrated how GANs can be applied for anomaly detection. GANs usually take a huge number of iterative steps to achieve promising results. Future work can include ensemble models to achieve better accuracy and performance within less number of training steps.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Samet Akcay, Amir Atapour-Abarghouei, and Toby P. Breckon. 2018. GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training. (Nov. 2018). https://arxiv.org/abs/1805.06725

[3] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly Detection: A Survey. *ACM Comput. Surv.* 41 (07 2009). https://doi.org/10.1145/1541880.1541882

[4] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. 2017. Adversarial Feature Learning. (April 2017). https://arxiv.org/abs/1605.09782

[5] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. 2017. Adversarially Learned Inference. (Feb. 2017). https://arxiv.org/abs/1606.00704

[6] Abhilash Garimella. 2020. *GANs in Real World - Can Bad Actors Use GANs to Beat AI?* Retrieved Feb 1, 2022 from https://bolster.ai/blog/gans-in-real-world-can-bad-actors-use-gans-to-beat-ai/

[7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. (Jun 2014). https://arxiv.org/abs/1406.2661

[8] Federico Di Mattia, Paolo Galeone, Michele De Simoni, and Emanuele Ghelfi. 2021. A Survey on GANs for Anomaly Detection. (Sept. 2021). https://arxiv.org/abs/1906.11632

[9] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. (Nov. 2014). https://arxiv.org/abs/1411.1784

[10] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. (Jan. 2016). https://arxiv.org/abs/1511.06434

[11] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. 2017. Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. (March 2017). https://arxiv.org/abs/1703.05921

[12] Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. 2019. Efficient GAN-Based Anomaly Detection. (May 2019). https://arxiv.org/abs/1802.06222