Signal, Image and Video Processing
https://doi.org/10.1007/s11760-022-02362-5

ORIGINAL PAPER

Cascade watchdog: a multi-tiered adversarial guard for outlier detection

Glauco Amigo[1] · Justin M. Bui[1] · Charles Baylis[1] · Robert J. Marks[1]

Received: 14 June 2022 / Revised: 2 September 2022 / Accepted: 11 September 2022
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2022

Abstract
The identification of out-of-distribution content is critical to the successful implementation of neural networks. Watchdog techniques have been developed to support the detection of these inputs, but the performance can be limited by the amount of available data. Generative adversarial networks have displayed numerous capabilities, including the ability to generate facsimiles with excellent accuracy. This paper presents and empirically evaluates a multi-tiered watchdog, which is developed using GAN-generated data, for improved out-of-distribution detection. The cascade watchdog uses adversarial training to increase the amount of available data similar to the out-of-distribution elements that are more difficult to detect. Then, a specialized second guard is added sequentially. The results show a solid and significant improvement on the detection of the most challenging out-of-distribution inputs.

Keywords Machine Learning · Cascade Watchdog · Autoencoder · GAN · Outlier Detection · Convolutional Neural Network

1 Introduction

Data augmentation is described as imagination or dreaming by C. Shorten and T. M. Khoshgoftaar [13]. They survey different methods of image data augmentation for deep learning, including adversarial training and generative adversarial networks (GANs) [5]. Adversarial training can be used to attack or defend systems, as well as to increase the amount of available training data. The goal of data augmentation is to create new data samples from the existing training set. This new data is obtained according to the purposes of the application [12]. In this work, data augmentation serves to supplement the kind of out-of-distribution outlier data that reside closest to the distribution manifold.

In the previous work [2,3], the autoencoder watchdog is introduced as a technique to identify out-of-distribution inputs to classification neural networks. The autoencoder watchdog combines input reconstruction with an error measurement calculation. This error is a measure of the distance between the input data and the training data manifold in the latent space.

Data samples distant from the manifold are determined to be out of distribution. As calculated error decreases, input data better resemble data close to the training manifold, and the more fuzzy the classification becomes. The work presented in this paper introduces the cascade watchdog, which specializes in identifying outliers data which resides close to the manifold of the distribution. This is achieved through a two-step process. First, training data are produced using a GAN, which has been trained to generate near-manifold images. Second, a binary classification neural network is designed to differentiate between in-distribution and near-manifold out-of-distribution data. The binary classifier is then added to the original watchdog data pipeline, creating a multilayered cascade watchdog. Additional layers may be added, as necessary, to achieve the desired performance.

2 Background

GANs are effective tools for data augmentation. Since the first publication, where Goodfellow et al. introduced GANs in 2014 [5], a variety of techniques and applications have been developed across diverse fields. Yi et al. [15] present a review of adversarial training in medical imaging, one of the

✉ Glauco Amigo
Glauco_Amigo1@Baylor.edu

✉ Justin M. Bui
Justin_Bui@Baylor.edu

[1] Baylor University, Waco, TX, USA

Published online: 23 September 2022

most prolific fields of application of GANs on data augmentation.

The basic structure of a GAN consists of a generator and a discriminator. The generator is trained to fool the discriminator, while the discriminator is trained to differentiate between real and fake inputs. After a GAN has been trained, it provides a source of freshly generated data samples, which resemble real system inputs. Once the generator of the GAN is trained, it produces in-distribution samples from noise. Variations of GANS and their many applications are available in the literature [4,17,18].

This paper presents a method of adversarial training inspired by the GAN model. The main contrast between the generative adversarial training method presented here and other widespread applications of GANs resides in the different target. Normally, the goal of a GAN is to generate data in the distribution of the dataset. However, in this paper, the goal is to obtain out-of-distribution samples within a certain distance of the distribution.

This work expands upon the concept of autoencoder watchdog neural networks, previously identified as a technique for outlier identification in classification networks [2,3]. The cascade watchdog presented in this paper improves upon the precision of the outlier identification task, demonstrating improved outlier identification while reducing misidentifications. In a nutshell:

1. The autoencoder serves as the discriminator of the GAN module,
2. The GAN module generates out-of-distribution data samples close to the distribution manifold, and
3. A combination of in-distribution and generated out-of-distribution data is used to train the binary classifier.

The binary classifier specializes in identifying outliers that are closer to the distribution manifold. Data far within the manifold are easy to classify. Classification of data close to the manifold surface is more difficult. Applying the autoencoder and the binary classifier in sequential order improves outlier identification, while preventing the network from discarding in-distribution elements by mistake.

Other approaches are available for outlier identification. Atlas et al. [1] and Hwang et al. [6,7] identified manifold boundary points using neural network inversion [8,9,14]. Yu et al. [16] propose a method that identifies outliers that are only far from the distribution. Lee et al. [11] use a generator component to produce data samples on the boundary of the distribution. They train the classifier to assign less confidence to the classification of inputs on the boundary of the distribution. In order to obtain less confidence at the output of the classifier for 'boundary' inputs, they set the output target as the uniform distribution for 'boundary' inputs during the training process.

Our approach differs from the previously mentioned methods. The components of the cascade watchdog, the autoencoder and the binary classifier, are designed, trained, and deployed to optimize their effectiveness: The autoencoder is used in the loss function of the GAN, and the GAN serves to generate an augmented dataset specifically crafted to optimize the performance of the binary classifier so that, once deployed, the layers potentiate each other.

3 Methodology

The distribution manifold lies is a small portion of the input space. The autoencoder layer is capable of identifying many outliers, covering a significant portion of the input space. To determine the boundary between in-distribution and out-of-distribution, the autoencoder uses a threshold hyperparameter. When selecting the threshold, there is a trade-off between false negatives (out-of-distribution data samples that are not identified as outliers) and false positives (in-distribution data samples that are classified as outliers). A large threshold reduces the false-positive rate but also increases the false-negatives rate, whereas a small threshold reduces the false-negative rate, but increases the false-positive rate.

The second layer of the cascade watchdog is a fine-grained binary classifier that complements the autoencoder layer. The binary classifier specializes in identifying outliers that reside close to the distribution. The threshold of the autoencoder can be increased to reduce the false-positive rate relying on the additional layer of defense provided by the binary classifier, which decreases the false-negative rate safely. At the end of this process, both the false-positive and false-negative rates are reduced. More outliers are identified, and less in-distribution data are erroneously discarded.

3.1 Adversarial watchdog

The autoencoder also supports the development of a GAN, which generates out-of-distribution data samples close to the autoencoder threshold. For training the GAN, two goals are necessary to generate a rich dataset:

1. Producing data samples where the autoencoder produces an error similar to the threshold.
2. Distributing the generated data samples across the boundary of the manifold. To avoid the collapse of the GAN, each generated output depends on a point on the manifold taken from the in-distribution training set. Each generated data sample comes from one input on the training

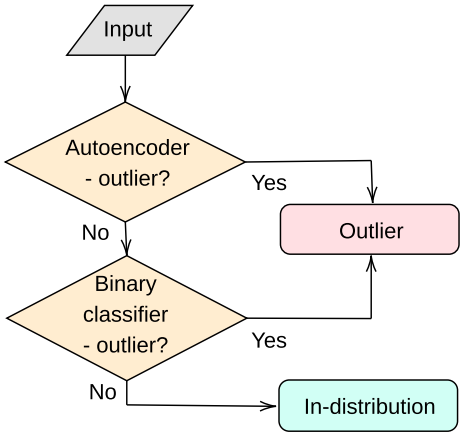


Fig. 1 Flowchart of the cascaded watchdog. An input that passes both layers of defense is considered as pertaining into the distribution

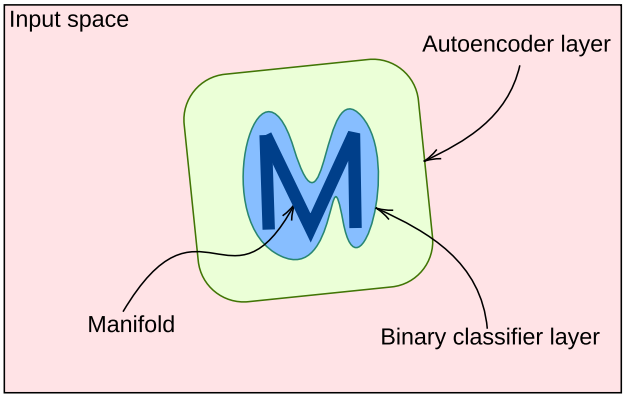


Fig. 2 Venn diagram dividing the input space. The biggest area is the set of outliers detected by the autoencoder, inside are those filtered by the binary classifier, and the bold "M" is the manifold with some false negatives around

data. The distance between generated outliers and inputs must be similar to the threshold of the autoencoder. The same error function between the input and output of the autoencoder is used to measure the distance between the generated out-of-distribution data sample and the original in-distribution data sample.

The combination of these two targets generates a dataset that is spread through the space near the boundary of the distribution manifold, preventing the collapse of the GAN.

The sequence of steps to produce the second layer of the cascade watchdog is:

1. Train the autoencoder.
2. Train the GAN and generate the dataset near the boundary of the distribution.
3. Create a fine-grained binary classifier.

The last step consists of training the binary classifier in two categories: in-distribution and out-of-distribution. The original training dataset is labeled as in-distribution, while the dataset near the boundary of the distribution generated with the GAN is labeled as out-of-distribution.

After both the autoencoder and binary classifier are trained, they combine sequentially to form the cascade watchdog, as seen in Figure 1. First, the input is analyzed using the autoencoder layer, which identifies whether the input is an outlier or not. If the autoencoder does not identify the input as an outlier, the input is then analyzed by the binary classifier. If neither the autoencoder nor the binary classifier identifies the input as an outlier, then the input is considered to be an in-distribution element.

The performance of the binary classifier is evaluated with a tenfold bias–variance analysis. The quality of the binary classifier is measured by observing the false and true negatives. Figure 2 shows a Venn diagram of the cascade watchdog formed by the autoencoder and the binary classifier. The domain of the autoencoder is the full input space and the outliers that are farther from the manifold are detected, while the domain of the binary classifier is only the space that the autoencoder does not filter. Ideally, all of the outliers are detected, while all in-distribution data are permitted. Observe that Figure2 does not contain false-positive outliers (i.e., no part of the manifold is marked as out-of-distribution), but has false-negative outliers (outliers that are very close to the manifold are not detected).

4 Experimentation

The first step of the experiments is to train the autoencoder with the MNIST dataset [1]. Examples of the MNIST dataset are shown in Figure 3. The GAN is trained to generate data samples close to the boundary of the distribution (see some examples in Figure 3).

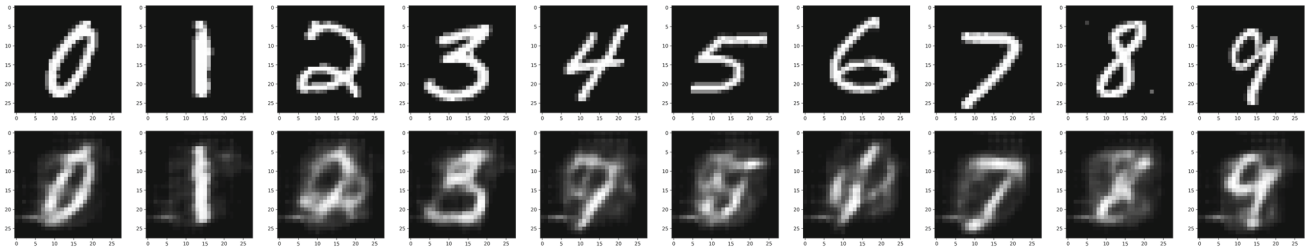
The boundary of the distribution is approximated by the threshold error function between the input and the output of the autoencoder. The error function, $\mathcal{L}_{\text{AE}}$, chosen for the threshold of the autoencoder is proportional to the root-mean-squared error (RMSE). The formula for the RMSE is:

$$\text{RMSE}\left(x, \hat{x}\right) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \hat{x}_i\right)^2}, \tag{1}$$

and the autoencoder error function is

$$\mathcal{L}_{\text{AE}}\left(x, \hat{x}\right) = \sqrt{\sum_{i=1}^{n}\left(x_i - \hat{x}_i\right)^2}, \tag{2}$$

[1] The MNIST dataset is available on TensorFlow: https://www.tensorflow.org/datasets/catalog/mnist.



(a) 4.2 **(b)** 2.1 **(c)** 7.3 **(d)** 4.9 **(e)** 5.8 **(f)** 5.6 **(g)** 7.4 **(h)** 4.8 **(i)** 5.9 **(j)** 4.3

Fig. 3 Illustration of the binary classifier training dataset. Each pair of images in the columns is an input and its corresponding output of the GAN. Images in the first row are original data samples from the MNIST dataset; the second row are the corresponding GAN-generated samples. The values in the captions of each Subfigure are the $\mathcal{L}_{\text{AE}}$ errors between original and GAN-generated samples. The images included in this figure are selected from each class of the dataset at random

where n is the number of pixels of the images (for the MNIST images, $n = 28 \times 28 = 784$), x_i is a pixel of the original image, and $\hat{x}_i$ is the reconstruction of the pixel by the autoencoder. According to (1) and (2),

$$\mathcal{L}_{\text{AE}} = \sqrt{n}\text{RMSE}. \tag{3}$$

For the experiments, a value of $\mathcal{L}_{\text{AE}} = 5$ is selected for the autoencoder threshold.

The loss function of the GAN, $\mathcal{L}_{\text{GAN}}$, has four components:

$$\mathcal{L}_{\text{GAN}} = \frac{a + b + c + d}{4} \tag{4}$$

where

$$a = \mid T - \text{RMSE}(x, y) \mid$$
$$b = \max\{0, \text{RMSE}(x, y) - T\}$$
$$c = \mid T - \text{RMSE}(y, \mathcal{R}_{AE}(y)) \mid$$
$$d = \max\{0, \text{RMSE}(y, \mathcal{R}_{AE}(y)) - T\}$$

and

- $|.|$ is the absolute value.
- x is an input to the GAN, a data sample from the MNIST training dataset.
- y is the GAN output, a data sample on the boundary of the distribution generated by the GAN from the current input, x, that is being used for training.
- $\mathcal{R}_{AE}(y)$ is the reconstruction that the previously trained autoencoder produces from the output of the GAN y.
- T is a threshold hyperparameter for the training of the GAN.

T is a different variable from the autoencoder threshold, even when they are technically related. T is an hyperparameter of the GAN that is fine-tuned for the GAN to generate images which the autoencoder will reconstruct with error close to the autoencoder threshold.

To get images close to the autoencoder threshold, $\mathcal{L}_{\text{GAN}}$ penalizes manifold distances that are greater or smaller than the threshold T. Distances to the manifold that are greater than T (included in components a, b, c, and d) are penalized double than those that are smaller than T (only included in a and c), because the former added twice as much as the latter in (4). This gives stability to the training process. At the same time, $\mathcal{L}_{\text{GAN}}$ is designed to satisfy the two goals of the GAN stated in Sect. 3.1: Components c and d serve to the purpose of Goal 1, by comparing the reconstruction error of the augmented dataset with the threshold T, and components a and b in (4) serve to the purpose of Goal 2, by comparing each output of the GAN y to a particular data sample in the manifold x.

Finally, the architecture of the binary classifier is shown in Figure 5. The binary classifier is a standard convolutional neural network classifier.

A geometrical parallelism can help to explain how $\mathcal{L}_{\text{GAN}}$ in (4) works. The GAN considers each data sample of the training dataset to generate another one at a small distance from the manifold. For training purposes, the generated data sample is projected back to the manifold with the autoencoder. The training goal of $\mathcal{L}_{\text{GAN}}$ is to make the distance from x to y equal to the distance from y to $R_{AE}(y)$, and equal to T. Intuitively, the GAN generates y from x in an orthogonal direction from the manifold, at a distance T (x lies on the manifold). This geometrical idea serves as inspiration for $\mathcal{L}_{\text{GAN}}$ in (4).

The results of the experiments show that this methodology to train the GAN produces diverse images at the desired distance from the manifold. To generate images with a specific $\mathcal{L}_{\text{AE}}$ value, the training of the GAN requires a smaller value for the threshold hyperparameter T. This is consistent with the multiplicative factor $\sqrt{n}$ in (3). For instance, the GAN needs $T \approx 0.2$ to obtain results of $\mathcal{L}_{\text{AE}} \approx 5.25$ (according

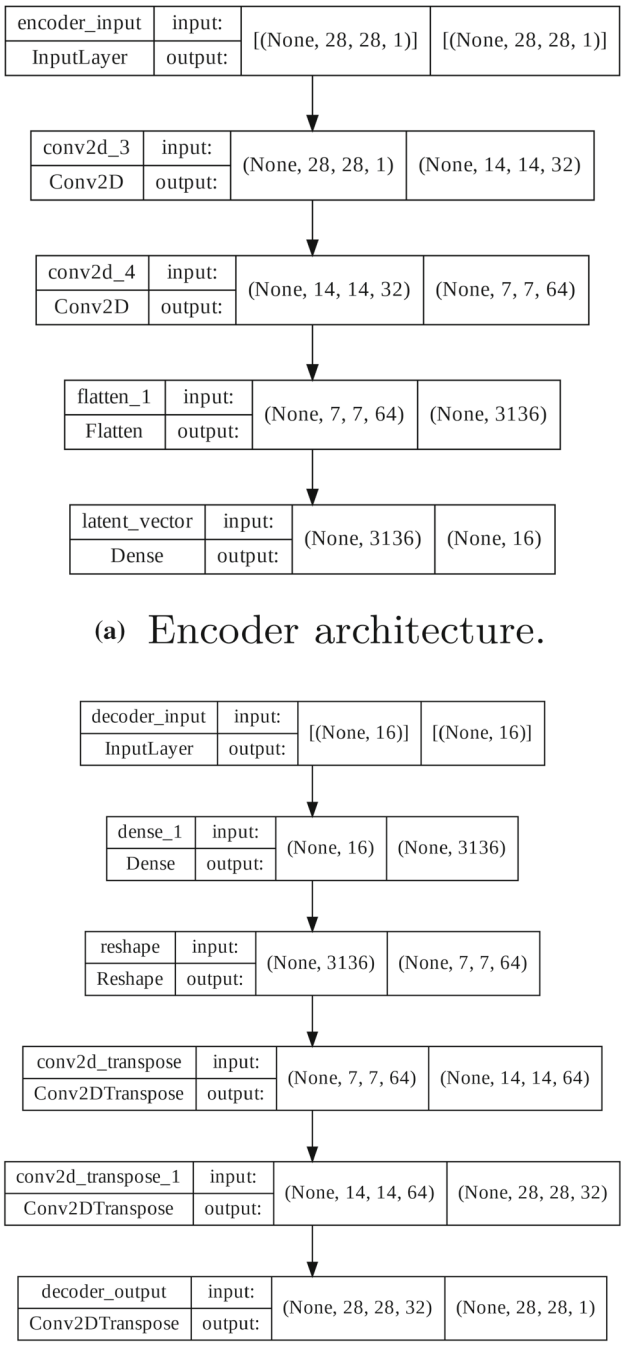


Fig. 4 Architecture of the autoencoder. The same structure is used for the autoencoder and for the GAN. Also, the GAN uses transfer learning from the autoencoder for the encoder and only trains the decoder

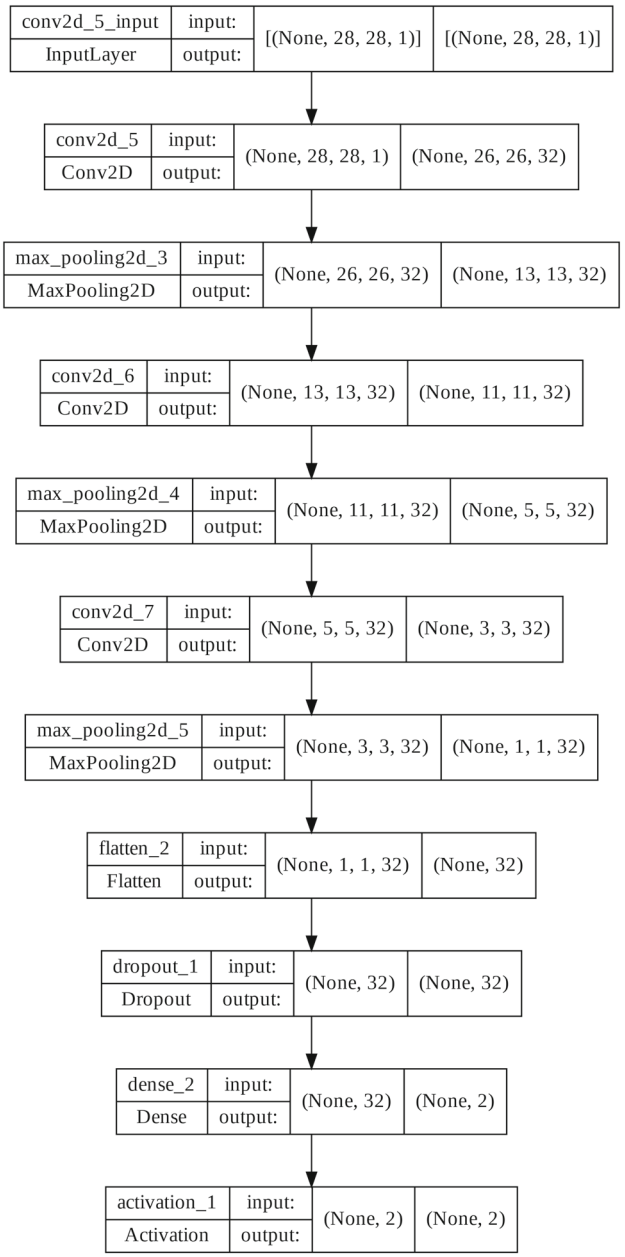


Fig. 5 Architecture of the binary classifier used as the second layer defense of the cascade watchdog

to (3), $0.2\sqrt{784} = 5.6$); a GAN threshold of $T = 0.05$ produces images with $\mathcal{L}_{\text{AE}} \approx 1.3$ (which is also relatively close to $0.05\sqrt{784} = 1.4$). Setting the hyperparameter $T \gtrapprox 0.3$ makes the GAN collapse. Using grid search [10], the optimal is $T = 0.1375$. With this value of T, the GAN produces images with an average $\mathcal{L}_{\text{AE}}$ of 4.14, which is below the threshold of the autoencoder value, set as 5, but it produces the best results at the end of the experimentation pipeline.

After training the GAN, one data sample on the boundary of the distribution is obtained for each input of the original dataset (see Figure 3). The training dataset of the fine-grained binary classifier layer consists of both the original in-distribution data and the generated near-the-boundary data.

The training of the autoencoder is excluded from the bias–variance analysis. Once trained, the autoencoder is used for the 10 experimental runs. On each run, the GAN is

Table 1 Tenfold bias–variance analysis for outlier detection of the binary classifier. Note: The first column corresponds to the threshold applied to the output of the softmax function on the binary classifier

In-distribution certainty #	True positive		False positive	
	Avg.	Std. Dev.	Avg.	Std. Dev.
1.0	0.77044	0.15077	0.03032	0.04879
0.99999999	0.77044	0.15077	0.03032	0.04879
0.9999999	0.74089	0.15728	0.01834	0.03079
0.999999	0.69939	0.16655	0.00752	0.01301
0.99999	0.63721	0.17212	0.00185	0.00320
0.9999	0.56401	0.15985	0.00030	0.00062
0.999	0.50184	0.13965	0.00006	0.00015
0.99	0.44579	0.11334	0.00001	0.00003
0.9	0.39786	0.09438	0.00000	0.00000

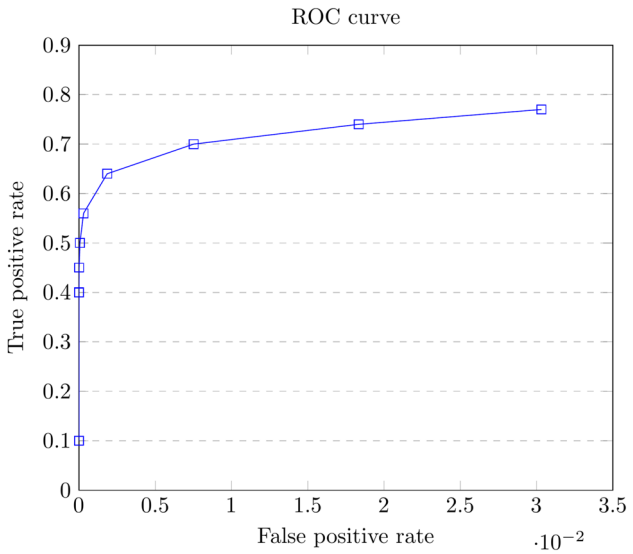


Fig. 6 ROC curve for outlier detection by the binary classifier (see Table 1). The point (1, 1) and the diagonal, which would represent random guess, are not included because they do not fit into the plot

trained reinitialized, allowing it to generate a fresh near-the-boundary dataset; then, the binary classifier is zeroed and then retrained. 50 000 data samples from the MNIST dataset are used in this process.

In order to characterize the performance of the cascade watchdog, 10 000 samples from the MNIST dataset and 10 000 outlier samples are used. The outlier samples are taken from the Fashion MNIST dataset [2]. Samples of the Fashion MNIST dataset can be seen in Figure 7. The in-distribution samples from the MNIST dataset used for testing are separate from the samples used for training.

The architecture of the autoencoder (see Figure 4) has an encoder and a decoder connected sequentially. In between, the latent space has 16 features. The architecture of the GAN is the same as the autoencoder. Transfer learning is applied for the encoder block of the GAN by copying the weighs from the autoencoder. The encoder of the GAN is then frozen during the training, and only the decoder block of the GAN is trained. In essence, the autoencoder latent representation of the MNIST training dataset is used as input for the GAN decoder. Or, in other words, the generative component of the GAN is trained to produce on-the-boundary data samples from the autoencoder latent representation of the in-distribution data samples.

5 Analysis

The workflow of the cascade watchdog (see Figure 1) has two steps:

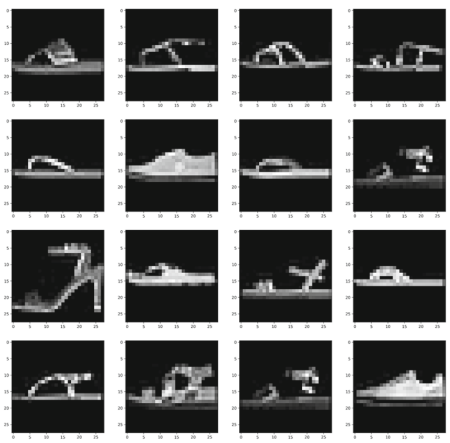
1. Autoencoder outlier detection.
2. Binary classifier outlier detection.

In the first step, the autoencoder detected 9347 outliers out of 10 000 data samples taken from the fashion MNIST dataset. In the second step, the remaining 653 outliers not detected by the autoencoder are tested on the binary classifier. The results of the tenfold bias–variance analysis are shown in Table 1 and in Figure 6. In Figure 6, the true-positive rate accounts for the fraction of the outliers that the binary classifier detected, while the false-positive rate corresponds to the in-distribution samples that the binary classifier layer classifies as outliers. The ROC curve characterizes a very good performance of the binary classifier, where the true-positive rate increases to about $\frac{3}{4}$, while the false-positive rate stays low. The specific values used to create this ROC curve are shown in Table 1. The binary classifier has a true-positive rate of 39.8% with 9.5% standard deviation, while preserving a zero false-positive rate. The false-positive rate is 0 until the threshold of the binary classifier certainty is raised. Approaching the certainty threshold to the limit (certainty

[2] The Fashion MNIST dataset is available on TensorFlow: https://www.tensorflow.org/datasets/catalog/fashion_mnist.

(a) Outliers detected by the autoencoder layer.

(b) Outliers detected by the binary classifier layer.



(c) Outliers not detected.

Fig. 7 Examples of fashion MNIST outliers detected by the autoencoder (Subfig. 7a), the binary classifier with a certainty threshold of 0.5 (Subfig. 7b), and not detected (Subfig. 7c)

greater than 0%) produces a small value for the false-positive rate (3%), while the true-positive rate improves from 40% to 77% in average.

In the experiments, the TPR of the standalone autoencoder is 93.47%. Considering a TPR of 77% for the binary classifier, the proportion of outliers that are identified on the system is $0.9347 + (1 - 0.9347) \times 0.77 = 0.985981$. Therefore, the combined TPR of the cascade watchdog is 98.6%.

In Figure 7, observe the differences between true-positive outliers filtered by the autoencoder and by the binary classifier. Also compare the detected outliers with the false negatives (unfiltered outliers). The outlier images not detected by the cascade watchdog, such as those seen in Figure 7c, resemble features from the in-distribution MNIST data: Angles, ovals, and writing strokes (see Figure 3). Between the detected outliers in Figures 7a and 7b, the differences with the MNIST data are more evident. The outliers detected by the binary classifier (see Figure 7b) are visually different from the MNIST data, but they also can be perceived as having some subtle features in common. On the other hand, between the outliers detected by the autoencoder (see Figure 7a), it is rare to observe features similar to the in-distribution MNIST dataset.

6 Conclusion

The cascade watchdog improves the trade-off between true and false negatives of the stand-alone autoencoder. Adding the binary classifier improves the true-positive rate, while reducing the false-positive rate. The enhancement of the outlier detection is possible due to the production of an augmented dataset by means of adversarial training. The autoencoder, the GAN, and the binary classifier work in conjunction to produce successful results.

The results of the experiments show that the binary classifier constitutes a significant contribution for out-of-distribution identification, encouraging the application of the binary classifier together with the autoencoder in future implementations. The high true-positive rate obtained in the experiments, combined with the low false-positive rate, confirms that the idea of splitting the out-of-distribution space into different subsets is a good approach for the detection of outliers. While the first layer defense (autoencoder) is capable of detecting most of the outliers, the second layer defense specializes in the out-of-distribution subspace that is closer to the manifold of the distribution. Adversarial training is capable of generating an augmented dataset to train the binary classifier. The cascade watchdog multi-tiered adversarial guard passes the proof of concept stage successfully and has the potential to be applied more broadly to real-world classification problems, which require the system to identify out-of-distribution inputs.

Acknowledgements Not applicable.

Author Contributions All the authors made substantial contributions to the conception and design of the work. G.A. and J.M.B. implemented and drafted the work, and C.B. and R.J.M. revised it critically for important intellectual content.

Funding Not applicable.

Data availability Employed datasets are linked in the manuscript

Declarations

Competing interests Not applicable

Ethics approval Not applicable

Consent to participate Not applicable

Consent for publication All authors agreed on the final approval of the version to be published.

References

1. Atlas, L.E., Cohn, D.A., Ladner, R.E.: Training connectionist networks with queries and selective sampling. In: Advances in neural information processing systems, pp. 566–573 (1990)
2. Bui, J., Marks, R.: Autoencoder watchdog outlier detection for classifiers. In: Proceedings of the 13th International Conference on Agents and Artificial Intelligence (2021) https://doi.org/10.5220/0010300509900996
3. Bui, J., Marks, R.: Symbiotic hybrid neural network watchdog for outlier detection. In: 17th International Conference on Machine Learning and Data Mining, MLDM 20, New York, pp. 171–180 (July 18–22, 2021)
4. Frid-Adar, M., Diamant, I., Klang, E., et al.: Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. Neurocomputing **321**, 321–331 (2018). https://doi.org/10.1016/j.neucom.2018.09.013
5. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., et al.: Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2. MIT Press, Cambridge, MA, USA, NIPS'14, pp. 2672–2680 (2014)
6. Hwang, J.N., Choi, J.J., Oh, S., et al.: Query learning based on boundary search and gradient computation of trained multilayer perceptrons. In: 1990 IJCNN International Joint Conference on Neural Networks, IEEE, pp 57–62 (1990)
7. Hwang, J.N., Choi, J.J., Oh, S., et al.: Query-based learning applied to partially trained multilayer perceptrons. IEEE Trans. Neural Netw. **2**(1), 131–136 (1991)
8. Jensen, C.A., Reed, R.D., El-Sharkawi, M.A., et al.: Location of operating points on the dynamic security border using constrained neural network inversion. In: Proceedings of International Conference Intelligent Systems Applications to Power Systems, (ISAP'97) (1997)
9. Jensen, C.A., Reed, R.D., Marks, R.J., et al.: Inversion of feedforward neural networks: algorithms and applications. Proc. IEEE **87**(9), 1536–1549 (1999)
10. LaValle, S.M., Branicky, M.S., Lindemann, S.R.: On the relationship between classical grid search and probabilistic roadmaps. Int. J. Robot. Res. **23**(7–8), 673–692 (2004)
11. Lee, K., Lee, H., Lee, K., et al.: Training confidence-calibrated classifiers for detecting out-of-distribution samples. Science **1711**, 9325 (2018)
12. Reed, R., Marks, R.: An evolutionary algorithm for function inversion and boundary marking. In: Proceedings of 1995 IEEE International Conference on Evolutionary Computation, vol. 2, pp. 794–797 (1995) https://doi.org/10.1109/ICEC.1995.487487
13. Shorten, C., Khoshgoftaar, T.: A survey on image data augmentation for deep learning. J. Big Data **6**, 1–48 (2019)
14. Thompson, B.B., Marks, R.J., El-Sharkawi, M.A., et al.; Inversion of neural network underwater acoustic model for estimation of bottom parameters using modified particle swarm optimizers. In: Proceedings of the International Joint Conference on Neural Networks, IEEE, pp. 1301–1306 (2003)
15. Yi, X., Walia, E., Babyn, P.: Generative adversarial network in medical imaging: a review. Med. Image Anal. **58**(101), 552 (2019). https://doi.org/10.1016/j.media.2019.101552
16. Yu, Q., Aizawa, K.: Unsupervised out-of-distribution detection by maximum classifier discrepancy. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 9517–9525, https://doi.org/10.1109/ICCV.2019.00961 (2019)
17. Zhang, H., Huang, Z., Lv, Z.: Medical image synthetic data augmentation using gan. In: Proceedings of the 4th International Conference on Computer Science and Application Engineering. Association for Computing Machinery, New York, NY, USA, CSAE (2020) https://doi.org/10.1145/3424978.3425118
18. Zhu, X., Liu, Y., Li, J., et al.: Emotion classification with data augmentation using generative adversarial networks. In: PAKDD (2018)

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

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.