Outlier detection using GAN

Sachin Vernekar
Dept. of Computer Science
University of Waterloo
sverneka@uwaterloo.ca

***Abstract*—The PAC learning theory [3] gives guarantees on classification accuracies, if the test data comes from the same distribution as the data which the classifier was trained on. So when a classifier sees an input sample that doesn't come from the training distribution, most likely the classifier is going to mis-classify the sample into one of the known classes, may be with a high confidence. We propose methods to detect these samples that don't belong to the training distribution p_{data} using a generative adversarial network (GAN) [6] and also propose a novel loss function to train the classifier with adversarial samples. We compare our methods with the baselines and show that our method significantly outperforms these baselines.**

The code for the experiments done is available at [21]

***Index Terms*—Outlier detection, GAN, MNIST**

I. Introduction

Usually in a classification network, we have a fixed number of output classes. For example, when training a classifier on MNIST dataset, we have 10 classes. So given any test image, the classifier tries to classify the image into one of the 10 classes depending on the class prediction probabilities. If we try to classify an input image that doesnt belong to any of the known classes, the classifier would still have to classify the image into one of the know classes, may be with a very high confidence. The problem of detecting such inputs that dont belong to any of the known classes is called open-set recognition [11].

PAC learning theory states that for a model with a given complexity (model complexity, roughly equal to number of trainable weights in a neural network), if we have trained the model with sufficiently large number of samples, sampled from the underlying distribution, p_{data}, it can guarantee with an arbitrary confidence, that the true error of the model would be arbitrarily close to the best error possible. But the assumption here is that the data that the model encounters, be it train or test, is sampled from p_{data}. If the data is sampled from outside of p_{data}, PAC won't be able to give any guarantees with respect to the classification accuracies.

Also in the case of adversarial attacks [4], especially for image classification, even though the input image looks just like noise, it is structured in such a way that it makes the classifier to classify it into one of the known classes with a very high confidence, say with 0.99 probability. In an another type of adversarial attack, an input image from a known class is modified by adding a structured noise in a way that the changes are not apparent to the human eyes, but the classifier misclassifies the image with a high confidence. One of the reasons why this happens in a traditional classifier is due to its nature of defining classification boundaries that are not bounded [4], and also to some extent due to over-generalization [4] [10]. But here too, we note that the adversarial examples come from outside of the training distribution.

It becomes imperative from the above discussion that the classifier works the best, when the input is sampled from the training distribution. Hence given an input, we need to first check if it belongs to the training distribution, and if yes, we trust the classifier output, else we say "we don't know" about the input.

Saying "we don't know" about an unknown input is very important in many machine learning applications, where safety is essential, such as autonomous driving. In literature, the problem of detecting such inputs is studied under the umbrella of outlier or anomaly detection. Broadly speaking, there are two ways to tackle this problem. Density based models [5] rely on estimating $p_{data}(x)$, given an input x. If $p_{data}(x)$ is large, then the input belongs to the training distribution, and if it is small, it would be detected as an outlier. But these models suffer from overestimation that leads to genuine inputs being classified as outliers. Also deploying these models in a classification setting would require more resources as these are deployed along with the classifiers.

Another approach is to do classification and outlier detection simultaneously. The most basic way to do this is by thresholding [7] where we threshold either the classification probabilities or the entropies of the classifier output distribution. Another way is to train an $n+1$ class classifier with adversarial examples as its $n + 1^{th}$ class. This is what is usually done in adversarial training [12], where the classifier is trained with adversarial examples generated by adversarial attack methods [4] [12]. But because the space of adversarial examples cannot be captured by any finite set of samples, the effectiveness of such an approach is limited.

II. Outlier detection using GAN

We propose a method to generate these adversarial examples using a GAN, and because in a GAN, the generator that generates samples that are far from p_{data} at the beginning of training, and slowly converges to p_{data} by the end of training, the hope is that the GAN, during the process of training would

produce optimal set of samples that in a way would form the boundary of data manifold. Hence if we train an $n+1$ class classifier with these adversarial examples as $n+1^{th}$ class, the decision boundary for n-classes is likely to be bounded, and anything that lies outside these n classes would be classified as $n+1^{th}$ class. Fig1 conceptually explains the idea.

There is already a paper [9] on this topic that was published recently while I was working on this idea. But there is a slight difference in our approaches. In their approach, they use the SSL-GAN [16] training approach, where the discriminator is replaced by an $n+1$ class classifier, where the $n+1^{th}$ class stands for fake data class generated by the generator and hence the classifier is trained along with the discriminator training. But they propose this design as the optimal outlier detector for a given false positive rate. This is because, during the process of training a GAN, the generator generates a mix of fake and true data; since the classifier is trained as a part of discriminator training, we train the discriminator to classify everything coming from the generator as fake which results in false positives in the classifier. In our "GAN approach", the $n+1$ class classifier is a separate entity from GAN and is trained offline. First, we train a GAN on the training data distribution, p_{data} and in the process of training, we generate few samples from the generator at each training epoch of the generator, we accumulate these samples until a point where the generator distribution p_g is close to p_{data}, then finally we pass all the accumulated samples through the discriminator to get the fake samples and take these samples to be the samples for $n+1^{th}$ class to train an $n+1$ class classifier.

There are few pros and cons with our approach compared to [9]. Our way of training the classifier will possibly solve the issue of false positives, but may be at the expense of introducing few false negatives depending on when we stop collecting fake samples from the generator. GAN training is usually unstable, this might lead to incorrect training of the classifier if trained in an online fashion and hence an offline training is preferred.

III. Outlier detection using GAN and attention loss

As discussed earlier, one of the ways to do outlier detection is to make the classifier give low prediction probabilities to the outliers, virtue of which, using a suitable threshold on the prediction probabilities, we will be able to detect the outliers. So we train an n class classifier using the outlier samples generated by the generator of the GAN minimizing a new loss function called *"attention loss"* applied only to outlier samples and normal cross-entropy loss applied to rest of the samples. The idea is that the *"attention loss"* tries to make the output prediction probabilities of outlier samples the least possible. Because we usually use a softmax layer at the output of the classifier, class prediction probabilities will sum up to 1. Hence, ideally for the outliers, we want the class prediction probabilities to be the same for all the classes and equal to $\frac{1}{number\ of\ classes}$. The *attention loss* is given by the equation below.

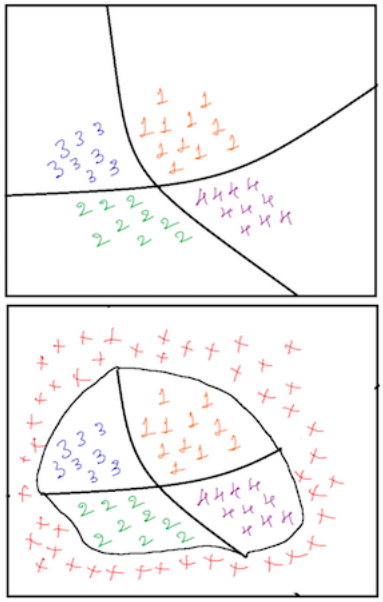

Fig. 1. Figure above shows a 4-class classifier with unbounded decision boundaries. Figure below shows the same classifier trained with "GAN approach" where 'x' represents generated/adversarial sample, resulting in bounded decision boundaries.

$$margin = \frac{1}{N}$$

$$attention\ loss = \frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{N} max(0, \hat{y}_{ij} - margin)$$

where N is the number of classes and M is the number of training samples in a batch and $\hat{y}_{ij}$ is the prediction probability for the j^{th} class and the i^{th} sample. The $attention\_loss$ attains the minimum value of 0 when all $\hat{y}_{ij}$'s equal $\frac{1}{number\ of\ classes}$. For example in case of an MNIST classifier, where we have 10 classes, we want all the class prediction probabilities for outlier samples to be equal to 0.1. *Attention loss* is very similar to *hinge loss* [1] that is often used in support vector machines. *Attention loss* has a constant gradient with respect to the prediction probabilities when the loss is positive that makes it a good loss for training deep neural networks. We would like to call the loss, *"attention loss"* because, we essentially want the network to pay attention to the real training data and discard the outliers. *Atttention loss* can be thought of as having a regularization effect on the network by shaping the decision boundaries to be bounded and hence reducing over-generalization.

The training procedure for "GAN with attention loss" is as follows.

1) Generate outlier samples using a GAN.
2) Train an n class classifier using the training data and outlier samples in siamese [2] style.
3) The model has 2 inputs and 2 outputs, and the model is used in a shared manner in siamese style. Hence each sample has 2 inputs and 2 outputs; the first input is

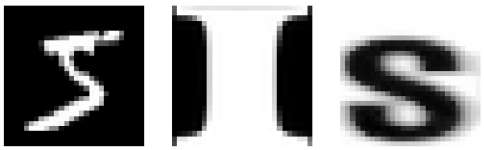

Fig. 2. Figure shows the contrast between MNIST, notMNIST, ICDAR-2003 images (from left to right respectively).

the normal training data sample with its corresponding labels, and the second input is the outlier sample with a dummy placeholder output.

4) For the first output, we apply the usual cross-entropy loss and for the second output, we apply *attention loss* with a constant scaling factor of λ which is a hyper parameter. Hence the total loss essentially is,

$$total\ loss = \text{cross-entropy} + \lambda \cdot attention\ loss \quad (1)$$

5) In all our experiments we set $\lambda = 0.5$.
6) We monitor both the losses during training and stop the training based on the validation accuracy of the first output.

IV. Experiments

The experiments were conducted on MNIST dataset, and we compared 3 methods of outlier detection, namely, "thresholding", "GAN method", and "GAN with attention loss" method. In all the 3 methods, a convolutional neural network with 3 convolutional layers was trained on MNIST dataset, whose architecture was kept the same. Training of both the classifier and GAN was done on MNIST training dataset containing 60K images of digits and 10K images from the test set were used for testing.

To get samples that don't belong to the training distribution, notMNIST [19] and ICDAR-2003 [20] OCR datasets were used. The notMNIST dataset contains 18K+ 28x28 gray-scale images of alphabets (A-J). ICDAR-2003 dataset contains 28x28 gray-scale 11K+ alphabets (A-Z&a-z) and 400+ digits. For our experiments we take only the alphabets and ignore the digits. MNIST images have a black background with the digit written in white pixels. Similarly notMNIST dataset contains alphabets written in white pixels with a black background. But ICDAR-2003 dataset is quite different in a way that both the background and the character are a bit blurry. Hence any optimum classifier that is trained on MNIST would tend to give bad results when tested on digits from ICDAR-2003 dataset without explicit domain adaptation. Fig 2 shows the contrast between images from 3 different datasets.

DCGAN [14] implementation of GAN was used. DCGAN works quite well on MNIST dataset. For more complex datasets such as CIFAR-10 and CIFAR-100, WGAN [15] is advised because it is experimentally found to be more stable on those datasets. DCGAN was trained for 5K epochs of both the generator and the discriminator training, and at each epoch, 10 images were randomly sampled from the generator and were saved. Finally, 50K saved images were passed through the discriminator to get 30K+ fake images to be used as outlier samples.

Outlier detection experiments were conducted separately on notMNIST and ICDAR-2003 datasets. For experiments on notMNIST, as test data, test data from MNIST was appended to notMNIST dataset, where notMNIST samples were given a label 10, indicating an outlier. Similar is the case for ICDAR-2003 dataset. The CNN classification accuracies on MNIST training and test data for each of the outlier detection methods are reported in Table I.

TABLE I
Table of classification accuracies on MNIST

Method	***Training Accuracy (%)***	***Test Accuracy (%)***
Thresholding	99.83	99.57
GAN Method	99.86	99.62
GAN with attention loss	99.69	99.57

A. Thresholding

As discussed earlier, thresholding refers to putting a threshold on the class prediction probabilities of the CNN and classifying everything that is less than a certain threshold as an outlier. In our case, the CNN is a 10 class classifier trained on MNIST training datset. Fig 4 and Fig 5 show the plot of classifier accuracies as a function of threshold ranging from 0 to 1. The maximum accuracy obtained are tabulated in Table II. As we can infer from the Table II that the classification accuracies drop down to abysmal $\sim 66\%$ from $\sim 99.5\%$ for notMNIST dataset. This shows that thresholding is not very effective in detecting the outliers. In case of ICDAR-2003 dataset, the results are still good. The accuracy is as high as $\sim 97\%$. This can be attributed to the nature of dataset discussed in the earlier section, where ICDAR-2003 dataset is drastically different from MNIST, making it easy to do outlier detection even with a naive method such as thresholding.

B. GAN Method

Here we train an 11 class CNN classifier where the 11^{th} class stands for outliers. The samples for the outlier class come from the generator of DCGAN trained. We then do a threshold sensitivity analysis as done in the thresholding method. From Table II, we can see a significant improvement over the thresholding method for notMNIST, where the maximum accuracy is $\sim 88.4\%$, an increase of $\sim 22.4\%$ over thresholding. Also

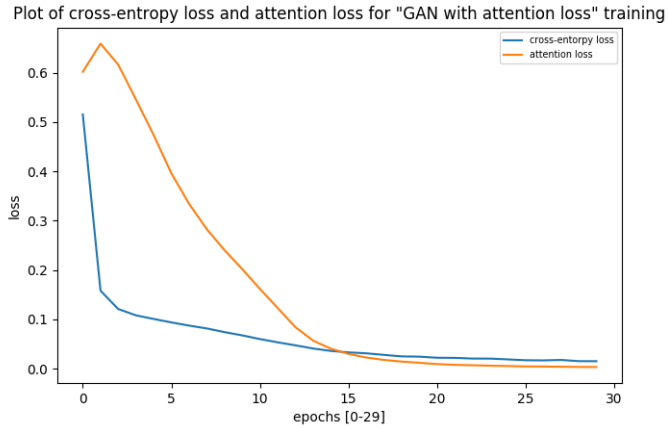

Fig. 3. Figure shows the plot of cross-entropy loss and attention loss for "GAN with attention loss" training.

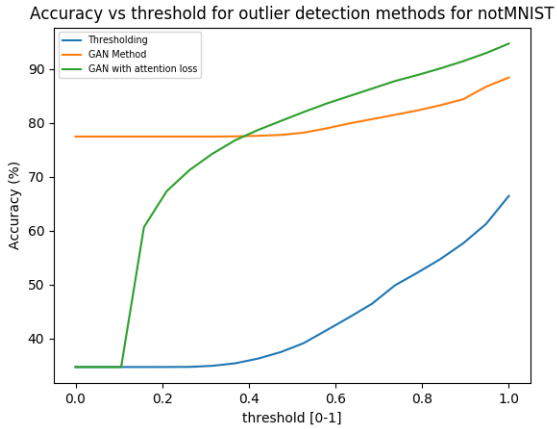

Fig. 4. Figure shows the plot of accuracies for different thresholds of prediction probabilities for notMNIST experiment.

for ICDAR-2003 dataset, the accuracy is as high as $\sim 99.5\%$. This proves the efficacy of GAN method as an efficient outlier detector.

C. GAN with attention loss

We train a 10 class CNN classifier with MNIST training dataset, where the first input is a sample from MNIST and the second input is the outlier sample sampled from the generator of DCGAN trained. Fig 3 shows the plot of both cross-entropy loss and attention-loss for training data. We can see that both the losses decrease as the number of epochs increase and go hand in hand. This shows that the new loss has a good gradient and doesn't introduce any instability in training. From Table II, we see that we get a maximum accuracy of $\sim 94.7\%$, which is an $\sim 8.3\%$ improvement over the GAN method for notMNIST. Also for ICDAR, the maximum accuracy is $\sim 99.7\%$ which again is the maximum among all 3 methods. With such stellar results from these limited set of experiments, GAN with attention loss method promises to be an efficient outlier detector.

TABLE II
TABLE OF MAXIMUM CLASSIFICATION PERCENTAGE ACCURACIES ON MNIST TEST+ OUTLIER DATASET

Outlier Set	Thresholding	*GAN Method*	*GAN with attention loss*
notMNIST	66.42	88.4	94.7
ICDAR-2003	96.92	99.5	99.69

V. DISCUSSION AND CONCLUSION

From the set of experiments conducted, it is evident that, both the "GAN" and "GAN with attention loss" methods of outlier detection are effective in detecting the outliers. But this approach heavily depends on the quality of outliers data generated by GAN. One of the assumptions we made was that, since the generator of the GAN tries to converge to p_{data} from outside of the support of p_{data}, before converging, it

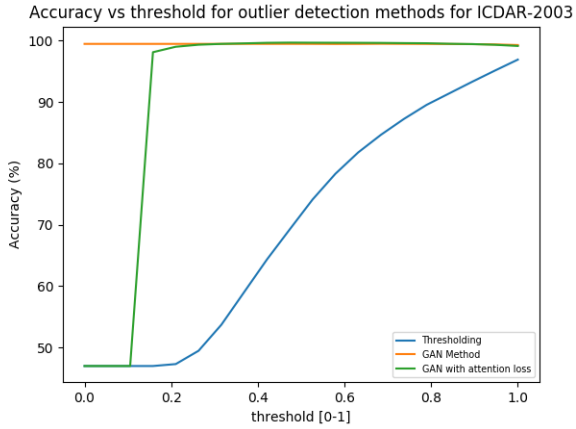

Fig. 5. Figure shows the plot of accuracies for different thresholds of prediction probabilities for ICDAR-2003 experiment.

produces data samples covering the entire boundary of the support. Thereby, when these data samples are used to train the classifier, it results in bounded decision boundaries for the required classes. This in fact is quite a strong assumption to make, hence even though practically the GAN approach works for outlier detection, we can't in theory justify that it works in all cases if trained optimally.

The experiments conducted are more or less proof of concept experiments. To gauze the practical efficacy of the proposed approaches, we would have to experiment with challenging datasets such as CIFAR-10, CIFAR-100 and possibly Imagenet. It would also be essential to compare the proposed approaches with more sophisticated outlier detection techniques based on density estimation, such as PixelCNN [17], PixelRNN [18], denoising auto-encoders [10], variational auto-encoders [13] etc. Density estimation methods such as PixelCNN, PixelRNN are more sophisticated because, in theory, if trained with models of enough capacity, it would tractably estimate the marginal density of the data distribution

$p_{data}(x)$. Hence outlier detection just boils down to checking if input x lies in the support of p_{data}. But due to the practical limitations discussed earlier, they are not very popular.

REFERENCES

[1] Katarzyna Janocha, Wojciech Marian Czarnecki
On Loss Functions for Deep Neural Networks in Classification. https://arxiv.org/pdf/1702.05659.pdf

[2] Sumit Chopra, Raia Hadsell, Yann LeCun Learning a Similarity Metric Discriminatively, with Application to Face Verification. CVPR 2005.

[3] Shai Shalev-Shwartz and Shai Ben-David
Understanding Machine Learning: From Theory to Algorithms Cambridge University Press Book, 2014, page 43-47.

[4] Ian J. Goodfellow, Jonathon Shlens and Christian Szegedy
Explaining and harnessing adversarial examples ICLR 2015.

[5] M. A. F. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko
Review: A review of novelty detection. Signal Process., 99, June 2014.

[6] Ian J. Goodfellow et.al.
Generative Adversarial Networks. NIPS 2014.

[7] D. Hendrycks and K. Gimpel.
A baseline for detecting misclassified and out-of-distribution examples in neural networks. ICLR, 2017.

[8] Yann LeCun et.al.
Energy-based Generative Adversarial Network. ICLR 2017.

[9] Mark Kliger, Shachar Fleishman.
Novelty Detection with GAN.
https://arxiv.org/pdf/1802.10560.pdf

[10] Giacomo Spigler
Denoising Autoencoders for Overgeneralization in Neural Networks. https://arxiv.org/abs/1709.04762

[11] Abhijit Bendale, Terrance Boult.
Towards Open Set Deep Networks. NIPS 2015.

[12] Ian J. Goodfellow et.al.
Explaining and Harnessing Adversarial Examples NIPS 2014

[13] Diederik P Kingma, Max Welling
Auto-Encoding Variational Bayes NIPS 2014

[14] Alec Radford, Luke Metz, Soumith Chintala
Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ICLR 2016.

[15] Martin Arjovsky, Soumith Chintala, Lon Bottou
Wasserstein GAN. ICML 2017.

[16] Augustus Odena
Semi-Supervised Learning with Generative Adversarial Networks ICML 2016

[17] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, Koray Kavukcuoglu.
Conditional Image Generation with PixelCNN Decoders. https://arxiv.org/pdf/1606.05328.pdf

[18] Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu.
Pixel Recurrent Neural Networks. ICML 2016.

[19] https://www.kaggle.com/lubaroli/notmnist

[20] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young.
Icdar 2003 robust reading competitions. ICDAR 03, 2003.

[21] https://github.com/sverneka/opengan