ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence
and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

GraN: An Efficient Gradient-Norm Based Detector for Adversarial and Misclassified Examples

Julia Lust[1,2] and Alexandru P. Condurache[1,2]

1- University of Lübeck - Institute for Signal Processing

2- Robert Bosch GmbH - Automated Driving

Abstract. Deep neural networks (DNNs) are vulnerable to adversarial examples and other data perturbations. Especially in safety critical applications of DNNs, it is therefore crucial to detect misclassified samples. The current state-of-the-art detection methods require either significantly more runtime or more parameters than the original network itself. This paper therefore proposes GraN, a time- and parameter-efficient method that is easily adaptable to any DNN.
GraN is based on the layer-wise norm of the DNN's gradient regarding the loss of the current input-output combination, which can be computed via backpropagation. GraN achieves state-of-the-art performance on numerous problem set-ups.

1 Introduction

Deep neural networks (DNNs) have proven themselves by achieving state-of-the-art performance in many application fields. Due to their good results, they are also used in safety-critical approaches, such as perception for autonomous driving [1]. Especially for such applications, it is crucial to understand the generalization performance of DNNs. While perfect generalization is difficult to achieve in practice, it is particularly important in situations where generalization issues are exploited on purpose. This is the case for adversarial examples [2], [3], [4], [5], i.e. small perturbations of the input, that are often unrecognizable to humans, but lead to a misclassification by the network.
There are two ways to tackle the problem of adversarial examples. One is to use *defense techniques*, which make DNNs robust against adversarial attacks by training. However, for each defense technique a new adversarial attack strategy can usually be found. To avoid this problem, other works build a method on top of the original network to *detect* adversarial examples [6], [7], [8], [9], [10], [11]. Unfortunately, the current state-of-the-art detectors require either significantly more parameters or more runtime than the original network itself. This is critical for applications like autonomous driving where the available resources are limited.
We therefore propose **GraN**, a parameter-efficient method whose main component is the **Gra**dient of the network with respect to the weights and the current input-output combination, more precisely its **N**orm (see Figure 1). GraN is based on the concept that misclassified examples have a larger gradient than

correctly classified examples. It can be calculated in a manner similar to the way a training step is conducted. Our experiments on three different datasets show that GraN can detect misclassified examples faster and with less parameters, while performing comparable to state-of-the-art detectors.

2 Related Work

Common adversarial detection methods can be split into three main categories [10]: denoisers, prediction inconsistency based approaches and metric based approaches.

Denoisers such as *MagNet* [7] and *HGD* [8] are based on a preprocessing procedure, in which the image is reduced to its main features by an encoder-decoder method. The idea is to reduce noise and the added adversarial perturbation. Despite their good performance, denoisers can not be used in parameter-restricted approaches since an additional encoder-decoder network with many parameters is required [8], [10].

Prediction inconsistency based approaches use the idea that for misclassified images the classification output is more sensitive to small changes in the image. The most successful approach is based on *feature squeezing*. The original image, an additional image with reduced color depth generated from the original image and a smoothed image are classified by the network. The probabilities for the three images are compared and the sensitiveness is used to decide whether the original image is misclassified [9]. Usually, only an insignificant number of parameters is used for such methods. However they do not perform well on some adversarial attacks [9], [10].

Metric based approaches use statistical methods to check whether the current input is behaving similar to normal inputs that have been investigated before. One of the best known is a detector based on the Local Intrinsic Dimensionality (*LID*) computed on the layer wise activation space [6]. It is a weighted distance metric based on the k-nearest-neighbors of 100 randomly chosen samples from the training set. This detector was outperformed by a method using the layer wise *Mahalanobis* distance to the closest class-conditional Gaussian distribution [11]. The most recent detector is based on Neural-network Invariant Checking (*NIC*) [10]. Additionally to the layer-wise activation distribution a classification network consisting of one fully connected softmax layer is trained on each activation layer. The more the class probabilities differ from layer to layer, the more probable the current example is misclassified. *LID*, *Mahalanobis* and *NIC* lead to good results but require a huge overhead of parameters or runtime in comparison to the actual classification network. In case of *LID*, 100 images need to be saved for comparison in the activation spaces. For *Mahalanobis* the mean and the covariance matrix for each class and often high-dimensional layer have to be stored and *NIC* additionally requires a huge overhead of parameters due to the fully connected network on top of each activation layer.

Although some detectors show compelling performance, none of the existing detectors meet the computational complexity restrictions of autonomous driving.

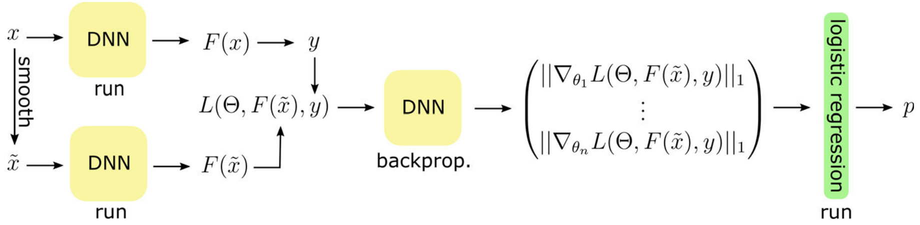
x
DNN
run
smooth
$\tilde{x}$
DNN
run
$F(x)$
y
$L(\Theta, F(\tilde{x}), y)$
$F(\tilde{x})$
DNN
backprop.
$||\nabla_{\theta_1} L(\Theta, F(\tilde{x}), y)||_1$
$||\nabla_{\theta_n} L(\Theta, F(\tilde{x}), y)||_1$
logistic regression
run
p

Fig. 1: Overview of GraN.

3 GraN: A Gradient-Norm Based Detector

In this section we first provide **GraN** as algorithm and secondly give information on the used strategies for the **indication of uncertainty**.

GraN is a detector predicting if a pre-trained DNN with weights Θ is misclassifying an input x. A flowchart of our approach is shown in Figure 1. The method starts by running the DNN for the current input x. As usual for the classification, the index with the largest value of the output $F(x)$ is defined as the class prediction y of the network. The same DNN is run for an input $\tilde{x}$, which is generated by smoothing x with a Gaussian-kernel with standard deviation s. Both, the output $F(\tilde{x})$ on $\tilde{x}$ and the predicted class y are fed into the loss function $L(\Theta, F(\tilde{x}), y)$. Then, the gradient with respect to the weights $\nabla_\Theta L(\Theta, F(\tilde{x}), y)$ is computed time-efficiently via backpropagation as in the training phase. In the next step the set of gradients $\nabla_\Theta L(\Theta, F(\tilde{x}), y)$ are transformed into a feature vector: For each layer $i \in \{1, \dots, n\}$ of the network, the gradient regarding the set of its weights θ_i is replaced by its L_1 norm

$$\nabla_\Theta L(\Theta, F(\tilde{x}), y) = \begin{pmatrix} \nabla_{\theta_1} L(\Theta, F(\tilde{x}), y) \\ \vdots \\ \nabla_{\theta_n} L(\Theta, F(\tilde{x}), y) \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} ||\nabla_{\theta_1} L(\Theta, F(\tilde{x}), y)||_1 \\ \vdots \\ ||\nabla_{\theta_n} L(\Theta, F(\tilde{x}), y)||_1 \end{pmatrix}.$$

The resulting vector has length n depending on the numbers of layers in the DNN. As in reference [6], a logistic regression network with $n+1$ parameters is trained to predict the probability p of the input x to be misclassified.

The central idea and hence the main **indication of uncertainty** is the norm of the gradient of the loss function $\nabla_\Theta L(\Theta, F(x), y)$ computed for the network output $F(x)$ and the resulting class y. Mathematically, the gradient over a variable expresses the effect of small changes of that variable on the output. Therefore, the bigger the gradient, the bigger the corresponding uncertainty.

The difference between the network output $F(x)$ and the resulting one-hot class vector y is one of the most simple indications of uncertainty. This evidence is covered by the gradient $\nabla_\Theta L(\Theta, F(x), y)$, since the difference directly builds the gradient over the last layer and represents the starting point for the backpropagation. Hence, a huge distance between $F(x)$ and y directly increases all gradient values.

To further enlarge the gradient for misclassified examples, the smoothed input $\tilde{x}$ is used for the prediction $F(\tilde{x})$. This smoothing procedure is thought to remove the perturbations responsible for misclassification. Thereby we increase the difference between the class y predicted for x and the network output $F(\tilde{x})$ and hence the gradient $\nabla_\Theta L(\Theta, F(\tilde{x}), y)$.

4 Evaluation

We evaluate GraN on five adversarial attack methods and two additional misclassification tasks each applied on three datasets. To be able to directly compare the performance of GraN, we choose to implement LID as current state-of-the-art method with the least number of parameters as discussed in Section 2.

4.1 Experimental Setup

All experiments are performed on the MNIST [12], CIFAR-10 [13] and SVHN [14] datasets. All datasets consist of predefined test and training sets which we refer to as pre-train and pre-test sets, since for the detection additional test and training sets are required. For each dataset a DNN for the classification task is trained on the pre-training data and tested on the pre-test set. The DNNs are taken from reference [6]. For MNIST, a 5-layer Convolutional Neuronal Network (CNN) was used. It achieved an accuracy of 96.89%. A 12-layer CNN was trained for CIFAR and an accuracy of 87.47% was reached. For SVHN, a 6-layer CNN was trained and an accuracy of 90.29 % was observed.

Similar to [10] and [9], the *adversarial set-up* is built from the correctly classified images from the pre-test set. For each of the images, five adversarial images are generated by the attack methods and settings reference [6] evaluated on: FGSM [2], BIM-a, BIM-b [3], JSMA [4] and CW [5]. Only the adversarial examples that lead to misclassification are kept. Each attack method defines together with the correct classified images one dataset. Additionally, two set-ups are build that more closely resemble situations with real world sensor data. For the *noisy set-up*, the correct classified images from the pre-test set are perturbed with Gaussian noise, such that one half of the noisy images is misclassified. For the *wrong set-up*, the original complete pre-test dataset is taken including the misclassified examples. Each data set-up is split into a train (80%) and a test (20%) set such that each set contains an equal number of misclassified and correctly classified examples, excessive samples are deleted.

For GraN, the standard deviation for the Gaussian smoothing is set to $s = 0.4$ for all datasets. The number of neighbors for LID was set as determined in reference [6]: $k = 20$ for MNIST and CIFAR and $k = 30$ for SVHN. For each set-up, the detectors are trained on the generated training set to predict the probability for an input to be misclassified and tested on the generated test set. The performance metric is the AUC ROC score [%]. It is defined as the area under the receiver operating characteristic (ROC) curve, which plots the true positive rate over the false positive rate. The results are provided in Table 1.

D.	Met.	Cause for misclassification						
		FGSM	BIM-a	BIM-b	JSMA	CW	Wrong	Noisy
M	LID	**100.00**	**99.70**	**99.99**	**99.25**	**99.87**	**97.35**	**96.33**
	GraN	99.51	98.46	99.19	95.78	92.03	93.25	93.96
C	LID	**99.20**	82.24	**100.00**	98.81	99.20	87.25	81.74
	GraN	98.82	**84.70**	99.74	**99.42**	**99.90**	**89.30**	**89.21**
S	LID	**99.99**	87.26	**99.98**	97.76	97.34	83.49	80.81
	GraN	99.98	**91.94**	99.96	**99.07**	**99.91**	**91.86**	**87.75**

Table 1: Performance of GraN (own) and LID [6] based on AUC ROC score [%] on different set-ups for MNIST (M), CIFAR-10 (C) and SVHN (S) dataset.

4.2 Discussion

The AUC ROC performance for GraN is between 90% and 100% for almost all set-ups. Except for MNIST, GraN outperforms LID in five of seven set-ups. The slightly worse performance on MNIST can be lead back to the simplicity of the MNIST dataset. Even with many small changes the original number is still recognizable for humans, whereas the network is concentrating on a small number of features that in most cases are enough for classification. This makes it easy for adversarial attacks because small details can be easily changed and hence the gradient stays low. The more complex the dataset is, the more diverse feature information is saved within the network. Even if the main features are changed, there are several other features contradicting the false decision, which therefore leads to a higher gradient. In most application fields, the data is more complex than the simple MNIST dataset. This promises good performance of GraN on such.

We remove perturbations leading to misclassification irrespective if the input is adversarial or not, by means of the Gaussian low-pass filter as provided in Section 3. It was found empirically, that the usage of $F(\tilde{x})$ instead of $F(x)$ is leading to a performance improvement but is not essential for GraN's functionality. This perturbation removal procedure works very well in particular when the perturbations are similar to sensor noise. For autonomous driving applications this is a major use case. However, as our experiments show, it works as well for a larger class of adversarial perturbations.

The necessary resources for GraN and LID are compared using the example of the CIFAR dataset [13]: In addition to the parameters of the classification network, LID needs ($100 \cdot 32 \cdot 32 \cdot 3 + 66 =$) 307.266 parameters to save 100 images and 66 parameters for its logistic regression network. GraN only needs the 35 parameters for its logistic regression network and therefore 0.01% from that of LID. Although there was no particular attention payed to runtime optimization, the computation time of GraN was 0.5% from that of LID. There is the option to reduce the runtime of LID by directly saving the activations for each image, however this would lead to an even larger overhead of parameters for LID and the computation time of GraN would still be less than 50% of that of LID.

5 Conclusion and Future Work

We introduced and evaluated GraN, a novel, parameter-efficient method that detects adversarial examples as well as other misclassified examples. It is able to distinguish between noisy examples that are correctly and incorrectly classified. The time- and parameter-efficient performance for various adversarial attacks, especially on the more complex data sets, make GraN a promising method for several real-world scenarios. The perturbation-removal approach used here is geared towards sensor-noise. Even though this seems rather specific, our experiments show that it works well for various adversarial attacks. In future work, we will assess the performance of other perturbation-removal procedures than Gaussian smoothing and investigate the effects of replacing the norm and logistic regression part with more powerful procedures.

Acknowledgment: We thank Matthias Rath and Stephan Scheiderer for their valuable input.

References

[1] L. Gauerhof, P. Munk, and S. Burton. Structuring validation targets of a machine learning function applied to automated driving. In *SAFECOMP*, pages 45–58. Springer, 2018.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015.

[3] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *ICLR*, 2017.

[4] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. Berkay Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *EuroSP*, pages 372–387, 2016.

[5] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *SP*, pages 39–57, 2017.

[6] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. N. R. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *ICLR*, 2018.

[7] D. Meng and H. Chen. Magnet: a two-pronged defense against adversarial examples. In *CCS*, pages 135–147, 2017.

[8] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *CVPR*, pages 1778–1787, 2018.

[9] W. Xu, D. Evans, and Y. Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *NDSS*, 2018.

[10] S. Ma, Y. Liu, G. Tao, W. Lee, and X. Zhang. NIC: detecting adversarial samples with neural network invariant checking. In *NDSS*, 2019.

[11] K. Lee, K. Lee, H. Lee, and J. Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NIPS*, pages 7167–7177, 2018.

[12] Y. LeCun, B. E Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, pages 396–404, 1990.

[13] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[14] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.