Histogram method for adversarial example detection in neural-network-based text classification

GUILLERMO GURIDI AREF MORADI
gugm | amoradi @kth.se

January 15, 2020

Text classification models are been widely used for practical applications like automated online content moderation or classification. They are thus exposed to input from anyone on the internet, without filter. Adversarial attacks are malicious inputs for machine learning models with the objective of making the model fail at its task. Most research has been focused on adversarial attack detection for image recognition tasks, and, amongst others, a method consisting on looking at histograms of the model's internal data has been proposed as a countermeasure. This research looks into whether this method is effective in the context of text classification as well.

An existing demonstrated adversarial attack for neural network text classification, called deepWordBug, is expanded and used as a source of adversarial examples against with detection methods are implemented. One of the base classifiers for which it generates adversarial examples is modified to implement the histogram method, which tries to determine whether the input was an adversarial example at prediction time. The evaluation results for this method results are subpar compared to current state of the art techniques but still open the door to new possibilities in this field.

Introduction

Automatic machine learning models that perform classification tasks are entering our daily live lives continuously whether we notice it or not. For example, online communities have been getting stricter content policies that are enforced by automatic algorithms. These algorithms are tasked with going through the users input and deciding whether there is copyright infringement, harassment, spam, etc...

It is therefore crucial to ensure that the automated models that look out for us, are as reliant and stable as possible. This is where the study of adversarial attacks comes from. If an attacker is aware that an automated procedure is going to judge their input they might try to exploit it (without actually having to incur in any hacking or other illegal activities).

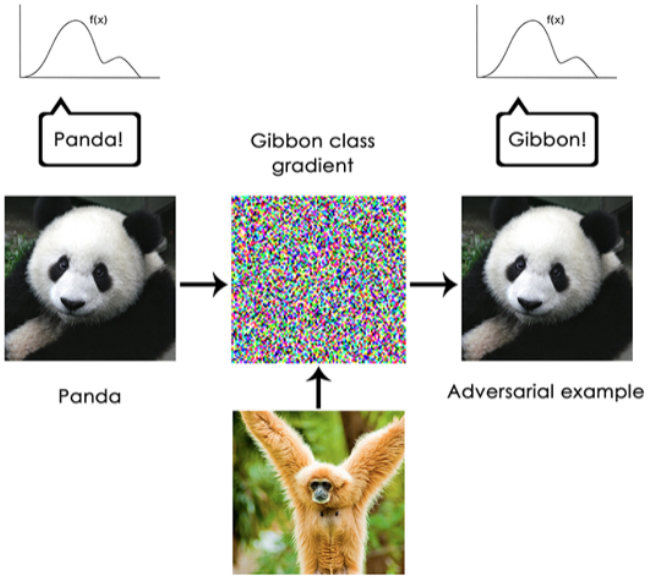

Image credit: François Chollet

Figure 1: Process of image alteration to have a model missclassify a panda picture..

These kind of attacks are called adversarial example attacks, because the only interface the attacker has with the model is through the input, and their goal is to try to conceal their input in a way that the classifier might fail to recognise their mischief. This is usually easy as models are generally trained with "clean" data in which the input that is to be flagged comes from actual examples from the real world, from a specific example distribution to say. But attackers might fabricate their own. These fabricated examples are usually small variations of actual inputs such that a human cannot tell the difference between them but they affect the targeted system in a malicious way, as showcased in figure 1 or 2.

There are already several different existing methods for generating such adversarial examples regardless of whether the attacker knows the techniques used for detection (white-box) or not (black-box) [1]. Many diverse methods for creating adversarial examples for image classifiers are mentioned in [1], [2] and [3], which give a good idea of the variety of methods that exist in the field and the complexity inherent to evaluating countermeassueres objectively. There are also diverse methods for creating adversarial examples for text classifiers, which are mentioned in [4] and [5]. The former is specially relevant since it is a black-box method oriented towards neural-network-base classifiers (current state of the art in text processing tasks), open-source, has been well documented by its authors and describes exactly which data sets it has been tested on. We can see an example of an adversarial example generated by this method in figure 2.

Parallel to this research, there have also been investigations into ways we can prepare the models for this abuse preemptively, either by making them more resilient or by detecting when they are being attacked [1]. These are two clearly distinct techniques:

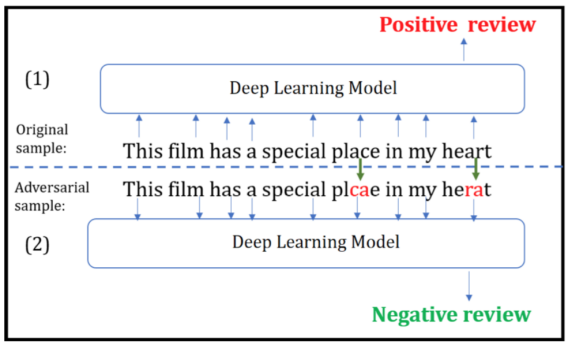

Figure 2: Example of a small modification to a text movie review that makes a classifier incorrectly predict the polarity of the review. Obtained from [5]

- The first of them happens before a model is trained. Adversarial examples have to be generated (sometimes using a previous traned version of the model) and those are reincorporated into the training phase to make the model more robust to such examples as can be seen in [6]. The main drawback of such methods is that they require much more training resources and modification of the existing model, sometimes not possible.

- The second techniques consists of attaching a second model to the original one, which is trained to detect adversarial examples. In this method the original model is not modified and this makes it suitable for protecting immutable models. Such examples can be seen in [7] and [8]

All the examples cited before have been made for image classifiers and cannot be directly applied to text classifiers. There is little research in detecting adversarial examples for text input and they are mainly focused on the first technique (retraining with adversarial examples) such as [9].

One recent paper [8] proposed three new methods for image adversarial example detection in the context of neural network classification, which haven't been implemented in text classification:

- **Regularization:** In an attempt to make the network more resilient to variations in small parts of the input, the last layer is regularized to avoid giving extreme weight to any of the individual contributions to the classifier outcome. This is achieved by freezing the weights of all the layers of the network but the last one, then the network is retrained with the same data but including a regularization term for the weights of that last layer, forcing it to be less reliant on single class indicators while keeping the accuracy of the network.

- **Histogram:** Going further inside the network, maybe we can look at the intermediate layers of the network and evaluate if it is working normally or not. This method proposes building a histogram of different layers of the network and feeding such histogram to a new model (in their case an SVM) which can decide if the network is functioning property (it has been given a normal example) or if it is being attacked

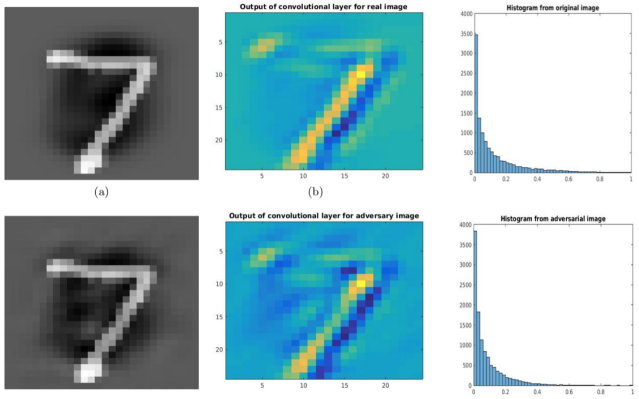

Figure 3: Input, internal layer and histogram from internal layer demonstrating the histogram method.

(it has been given an artificially modified example). A depiction of this process can be seen in figure 3.

- **Residual image:** They introduce the concept of the residual image, which contains information about the parts of the input pattern that are ignored by the neural network. This method aims at the detection of possible adversarial examples, by using the residual image and reinforcing the parts of the input pattern that are ignored by the neural network.

Choice of research method

The main purpose of this research is to give more insight into the possibility of porting adversarial example detection methods built specifically for image inputs to text inputs. As this topic alone would be too broad, we have to pin it down to a more narrow research problem.

From all the possible machine learning tasks and types of models, we are going to be looking at classifiers, which given an input will try to classify it into one of some pre-defined categories. Examples of such models range from image classifiers that try to identify the object depicted in the input, not-safe-for-work content detection, spam email or comment detection, etc... Adversarial attacks to regression models have been shown but evaluation of the effectiveness of countermeasures for them is significantly harder than with classifiers [10].

From all the proposed image adversarial example countermeasures that have not been tried in text yet we decided to focus first on the histogram method introduced in [8]. One of the main reasons to choose it was that we want to evaluate how well methods from images port to text, and histograms, of all other proposals, are the ones that seem specially targeted towards images. Histograms are commonly used to evaluate the exposure of images, or even as references to the color and luminance characteristics of an image [11]

and it is not clear exactly what the histogram of a text would mean (or of the internal representation of it as the method proposes). Nevertheless, it might still be useful in determining the presence of adversarial text examples as it has shown for images.

A direct consequence of this choice is that we restrict ourselves to neural-network-based models, as those are the ones for which the method has been designed, thanks to having their internal representations easily available. Thus, we had to look for a baseline neural network text classification task for which adversarial attacks had already been demonstrated. It turns out the research in [5] meets all those conditions and, as mentioned before, the entirety of its code is open source and freely available. As it contains many models and adversary example generation methods we chose the generator and model pair that showed best results (the model was not so easily deceived). Concretely, a simple embedding and simple lstm layer classifier with a character-swapping generator.

Thus, in the following sections we will look at how performant the histogram method (in its simplest form possible) is at detecting text adversarial examples generated via the character-swapping method from [5] in the aforementioned classification model also from the same paper.

To evaluate precisely how well our implementation works we will be calculating the average f1-score [12] of the two classes (adversarial / not adversarial) that our new model will be predicting. The choice of this metric comes from following the standard used in most of the literature that has been referenced here, and in an attempt to take into account the two types of mistakes our model can do: false-positive (normal input classified as adversarial) and false-negative (adversarial input classified as normal).

Application of research method

Data gathering

As we will be building a new SVM model using the histogram from the internal representation of an existing model, we need data to train it. To keep as close as possible to the research we are building on, we uses the same data-sets as they have. Luckily they are also documented in [5] and several download links are provided in their github repository. We have mainly focused in one of them, due to its simplicity. That is the imdb movie review polarity data-set, consisting on the text human-written reviews of thousands of movies, along with other metadata such as the star rating. The polarity in it comes from the fact that a binary label (positive or negative) has been inferred from the star rating of the movie, to make the classification problem simpler. The original data source can be found in [13].

Furthermore, we also need a data set of adversarial examples to test if our new model is able to tell them apart. We generate it by using the existing implementation in [5], using the original data as a basis for the new adversarial examples. Concretely, we took 5000 reviews, passed them through the adversarial generation processed and got a 500 successful adversarial examples. That is, texts that albeit looking very similar to the

original, managed to change the output of the classifier.

The histogram method

As mentioned in the previous section, the model that we will be trying to detect adversarial examples in comes already within the implementation of deepWordBug. It implements a typical text classification architecture with a simple RNN network. It consists of an embedding layer with the vocabulary size of 20000 and vector size of 100, one LSTM layer with embedding an hidden dimension of 100, a fully connected linear layer with 2 outputs and finally a softmax layer for 2 classes to provide the sentiment.

The histogram method consists of using the histogram from the network's internal layers to determine whether the input is adversarial. Trying to evaluate this approach in its simplest form, we only look at the ouputs of the LSTM layer and feed them into an SVM model. Thus, in the end we are interested only in the performance of this SVM model. To test it we create a training dataset from the aforementioned 500 reviews by joining them with their not adversarial counterparts and feeding them through the classifier, obtaining the required histograms. These histograms were created using 50 bins, giving a final dataset of 1000 50-dimensional vectors representing histograms.

The SVM model was trained using a linear kernel and the default parameters parameters provided by scikit learn, due to the reduced scope of this project, which didn't leave time for further model tuning.

Results and Analysis

The available data was divided into two sets consisting of 80% and 20% of the available data points, one for training and one for validation. During this division, the original texts' histograms and their adversarial counterparts were kept together so that both the training and evaluation phase would see the two variations. This makes the training more robust and keeps the result more relevant, avoiding information cross-contamination between training and evaluation.

Furthermore, we repeat this procedure randomly five times, effectively crossvalidating with the entire data set and averaging the metrics obtained afterwards. This procedure could also be repeated with more data sets to obtain more representative results.

Our preliminary results can be seen in table 1.

	precision	recall	f1-score	support
0	0.66	0.57	0.61	58
1	0.62	0.71	0.66	58
avg / total	0.64	0.64	0.64	116

Table 1: Scores

The results indicate that the model is slightly better than selecting the label by random which might be useful but is not optimal and is in no way up to state-of-the-art standards in two ways. On one side, the effectiveness for this histogram method shown in [8] is over 90% in normal normal settings. On the other hand, other adversarial text-classification prevention methods have also achieved higher accuracies (above 85%) albeit in different data-sets and settings [9].

Discussion

As discussed, our experiments don't reach anywhere close to the state of the art. Nevertheless, they haven't been proven to be totally useless either. It is good in this case to not forget all the limitations of our research that could have affected this result. The data gathered was probably insufficient for the task (barely a 1000 datapoints). It was also not diverse, as it would have been interesting to try with multiple data-sets. The model training was not performed up to current standards (meta-parameter optimization, more model types, etc...). No other classifier architectures were tried. Not all of the layers from the neural network were taken into account for generating the histogram. The number of bins for the histogram was kept fixed, which we don't know if may have an impact on the result. Also, there are several improvements applied on the histogram method in the source paper which were not applied here. And probably more things that would come up with further testing and more thorough work towards the goal of this project.

Due to the reduced dedication to this project, we didn't take the research to the full of our abilities and intentions, but we still believe methods used for detecting adverserial inputs for image classifiers can be potentially used for detecting adverserial input in their text counterparts. We would suggest the improvements on the current method to be tested first and further on trying with different techniques. We hope to have opened the door to more development towards a more common framework of adversarial example defenses between all types of inputs that machine learning models could have.

References

[1] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019. doi: 10.1109/TNNLS.2018.2886017

[2] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. doi: 10.1109/cvpr.2016.282. [Online]. Available: http://dx.doi.org/10.1109/cvpr.2016.282

[3] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016.

[4] Y. Li, P. Xu, and M. Pang, "Adversarial attacks on word2vec and neural network," in *Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence*, ser. ACAI 2018. New York, NY, USA: ACM, 2018. doi: 10.1145/3302425.3302472. ISBN 978-1-4503-6625-0 pp. 50:1–50:5. [Online]. Available: http://doi.acm.org/10.1145/3302425.3302472

[5] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," *CoRR*, vol. abs/1801.04354, 2018. [Online]. Available: http://arxiv.org/abs/1801.04354

[6] B. Biggio, G. Fumera, and F. Roli, "Multiple classifier systems for adversarial classification tasks," 06 2009. doi: 10.1007/978-3-642-02326-2_14 p. 132 to 141.

[7] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," 2017.

[8] S. Pertigkiozoglou and P. Maragos, "Detecting adversarial examples in convolutional neural networks," 2018.

[9] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," 2016.

[10] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 19–35.

[11] R. Maini and H. Aggarwal, "A comprehensive review of image enhancement techniques," *arXiv preprint arXiv:1003.4053*, 2010.

[12] C. J. Van Rijsbergen, *Information retrieval*. Citeseer, 1979.

[13] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text," in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 165–172.

This report has been elaborated by Aref Moradi and Guillermo Guridi in the context of the course Research Methodology and Scientific Writting (II2202) at KTH Royal Institute of Technology in Stockholm, Sweden. The supervisor for this work has been professor Henrik Boström, who works for the Division of Software and Computer Systems at KTH. The course run through the Autumn semester of the academic year 2019-2020, and ended with a presentation of the report the 10th of January 2020.