Exploring Gaps in DeepFool in Search of More Effective Adversarial Perturbations

Jon Vadillo[1,*], Roberto Santana[1] and Jose A. Lozano[1,2]

[1]University of the Basque Country UPV/EHU, 20018 San Sebastian, Spain
[2]Basque Center for Applied Mathematics (BCAM), 48009 Bilbao, Spain
[*]Corresponding author
Email: {jon.vadillo, roberto.santana, ja.lozano}@ehu.eus

Abstract

Adversarial examples are inputs subtly perturbed to produce a wrong prediction in machine learning models, while remaining perceptually similar to the original input. To find adversarial examples, some attack strategies rely on linear approximations of different properties of the models. This opens a number of questions related to the accuracy of such approximations. In this paper we focus on DeepFool, a state-of-the-art attack algorithm, which is based on efficiently approximating the decision space of the target classifier to find the minimal perturbation needed to fool the model. The objective of this paper is to analyze the feasibility of finding inaccuracies in the linear approximation of DeepFool, with the aim of studying whether they can be used to increase the effectiveness of the attack. We introduce two strategies to efficiently explore gaps in the approximation of the decision boundaries, and evaluate our approach in a speech command classification task.

Keywords: Adversarial examples, DeepFool, Robust machine learning.

1 Introduction

The intriguing vulnerability of Deep Neural Networks (DNNs) to imperceptibly yet maliciously perturbed inputs, known in the literature as adversarial examples [24], has raised concerns regarding the robustness of these models in adversarial scenarios, and more particularly in security-critical applications.

While different hypotheses have been proposed in the literature to explain why DNNs are vulnerable to such imperceptible perturbations, most of them focus on the analysis of the decision spaces learnt by the classifiers [6, 10, 13, 19, 24, 26]. Furthermore, the underlying theoretical framework of different attack strategies relies directly on wisely exploiting different properties of such decision spaces [14, 15, 16], which is the case of DeepFool [16], a state-of-the-art algorithm based on linearly approximating the decision boundaries of the target classifier to efficiently approximate minimal perturbations capable of inducing a misclassification.

Even outside the particular field of adversarial machine learning, the study of the decision boundaries of DNNs is currently a relevant yet understudied research topic [9, 13, 30], and advances in this direction are necessary to better understand the complex behaviour and decision making process in these models.

The objective of this paper is to study whether it is possible to exploit the inaccuracies in the linear approximation assumed in DeepFool in order to increase the effectiveness of the attack, while maintaining a minimal distortion. For instance, one could increase the number of incorrect output classes that can be produced in the model with a negligible overhead in the original algorithm, or find *shortcuts* to closer decision boundaries that are missed by DeepFool, reducing the amount of perturbation. From another point of view, the analysis of such inaccuracies in the linear approximation could also reveal interesting properties about the geometry of the decision space of the classifier, or provide a useful framework to study the proximity between the classes, which is directly related to the robustness of the classifiers to adversarial attacks.

For these purposes, in this paper we introduce two different methods for extending the DeepFool algorithm to efficiently explore inaccuracies in its linear approximation, and to study whether such inaccuracies can be exploited to generate more effective perturbations. Our experiments reveal that, although inaccuracies can be found in DeepFool, taking advantage of such inaccuracies does not result in a significant improvement over the original algorithm.

2 Related Work

The intriguing phenomenon of the vulnerability of DNNs to imperceptible adversarial perturbations was first reported by Szegedy et al. in [24]. Although multiple attack approaches [1, 2, 3, 6, 11, 12, 14, 16, 17, 23], and defensive strategies [5, 6, 7, 18, 21, 29] have been proposed for different tasks and domains, the explanation of these vulnerabilities, the connection between different attack strategies or the reason for common vulnerabilities on different models are still open questions.

Regarding the theoretical justification of adversarial examples, different hypotheses have been put forward. In [24], adversarial examples are attributed to the highly non-linearity nature of DNNs, causing dense low-probability *"pockets"* in the input space of the model, composed of inputs that, with very low probability, could be found by randomly sampling in the vicinity of a clean sample. Contrarily, in [6] it is stated that even linear models are also highly vulnerable to adversarial examples for high dimensional problems. In [22, 25], the existence of adversarial examples is studied under the manifold hypothesis: the data lies on a low-dimensional manifold $\mathcal{S}$ embedded in a high-dimensional representation of the input space, and, due to the high dimensionality of the input data, samples close to $\mathcal{S}$ can be found outside the decision boundary of the corresponding class. In [8], however, a data-perspective explanation is provided, explaining adversarial examples as *non-robust* features of the input data, instead of flaws in the learnt representation of the DNNs.

In this work, we focus on the exploitability of the geometry of the decision regions learnt by the DNNs as a basis for studying the adversarial examples [14, 15, 16]. In [16], the DeepFool algorithm was introduced, a state-of-the-art method for efficiently crafting adversarial examples, which is based on pushing an input to its closest decision boundary, approximated in the vicinity of the input according to an efficient linear approximation. A detailed explanation of this attack algorithm can be found in Section 3. In [14], this method has been extended in order to generate universal (input-agnostic) adversarial perturbations for the image classification task. In fact, the approach proposed in [14] is based on accumulating individual perturbations generated using the DeepFool algorithm. This strategy has also been used in the audio domain [27].

The theoretical framework introduced in [14] to generate universal perturbations is further developed in [15], where the authors study the relationship between the robustness of the models to universal adversarial perturbations and the geometry of their decision boundaries.

In this work, we intend to further exploit the decision boundaries of the target classifier to achieve more effective perturbations. In particular, we extend the DeepFool algorithm to expand its search space, and overcome the possible limitations that the assumed simplification of the decision boundaries may produce.

2.1 Technical Background

Let us consider a classification function $f : X \to Y$, being $X \subseteq \mathbb{R}^d$ the d-dimensional input space and Y a discrete output space of k classes, where $y_i \in \{y_1, \ldots, y_k\}$ represents the i-th class. Let $x \in X$ be an input sample correctly classified by f. The objective of an adversarial attack is to produce a perturbed input x' which, being highly similar to x, produces a misclassification of f, that is, $f(x) \neq f(x')$. To ensure that x' is as similar as possible to x, in this work we require the adversarial example to satisfy $\varphi(x, x') \leq \varepsilon$, where $\varphi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ represents a suitable distortion metric, typically an ℓ_p norm, and ε a maximum distortion threshold.

Depending on the malicious effect we want to produce on the classifier, we can consider different types of adversarial attacks. Given a clean input x, a *targeted* adversarial attack consists of perturbing x so that x' is wrongly classified as one particular target class $y_t \neq f(x)$. Contrarily, an *untargeted* adversarial attack consists of generating an adversarial example x' so that $f(x') \neq f(x)$, without any additional regard for the output class. A more comprehensive overview of the possible attack types can be found in [31].

3 Attack Algorithm

As previously mentioned, in this paper we use different variants of DeepFool [16], a state-of-the-art algorithm to generate adversarial attacks, initially introduced for images. Thus, in this section we provide a more detailed overview of this algorithm.

The objective of the attack is to find the minimal perturbation capable of sending an input sample x outside its decision region, by pushing it to the closest decision boundary. This can be seen as an optimization problem, in which we aim to find

$$r^* = \underset{r}{\operatorname{argmin}}\, \varphi(x, x+r) \;\; \text{s.t. } f(x+r) \neq f(x). \tag{1}$$

However, for high-dimensional non-linear decision spaces, which is the general case of DNNs, estimating the decision boundaries is a complex task, which makes this optimization intractable in practice. Due to this limitation, the DeepFool algorithm provides a strategy to approximate r^* by efficiently approximating the decision region of the model. This strategy consists of iteratively pushing an initial input x_0, of class $f(x_0) = y_c$, towards a linear approximation of the decision boundaries, based on the first-order derivatives in the vicinity of the input sample. This transforms the decision region into a polyhedron:

$$\widetilde{\mathcal{R}}_i = \bigcap_{j=1}^{k} \{x : f_j(x_i) - f_c(x_i) + \triangledown f_j(x_i)^\top x - \triangledown f_c(x_i)^\top x \leq 0\}, \tag{2}$$

where x_i represents the input sample at step i and f_j represents the output of f corresponding to the class y_j. Thus, the task of determining the minimal distances to such boundaries is now tractable. Based on this approximation, the closest decision boundary, which will correspond to a class y_l, is determined as follows:

$$l = \underset{j \neq c}{\operatorname{argmin}} \frac{|f'_j|}{||w'_j||_2}, \tag{3}$$

where $f'_j = f_j(x_i) - f_c(x_i)$, and $w'_j = \triangledown f_j(x_i) - \triangledown f_c(x_i)$ represents the direction of the perturbation. Finally, x_i is pushed towards the selected decision boundary according to the following rule:

$$r_i \leftarrow \frac{|f'_l|}{||w'_l||_2^2} w'_l \tag{4}$$

$$x_{i+1} \leftarrow x_i + r_i. \tag{5}$$

The algorithm stops when we finally produce a new incorrect class $f(x_i) \neq y_c$ in the target model.

As previously mentioned, we consider that the adversarial example should be restricted by a maximum amount of distortion ε in comparison to the original input, according to a suitable distortion metric: $\varphi(x, x') \leq \varepsilon$. Therefore, in this paper we assume that, if at any step of DeepFool the amount of distortion exceeds ε, then it is not possible to construct a *valid* adversarial example x' which is able to fool the classifier.

Finally, even if we mainly focus on untargeted attacks in this work, note that a targeted version of DeepFool can be obtained if the input is pushed towards the direction of the boundary corresponding to the target class $y_t \neq f(x_0)$ until $f(x_i) = y_t$ is satisfied. This can be easily achieved by removing the criterion specified in equation (3), and using the following update rule:

$$r_i \leftarrow \frac{|f'_t|}{||w'_t||_2^2} w'_t \tag{6}$$

$$x_{i+1} \leftarrow x_i + r_i. \tag{7}$$

4 Exploiting Gaps in the Linear Approximation of the Decision Boundaries

DeepFool relies on the assumption that we can accurately approximate the boundaries of the decision region of the classifier in the proximity of a given input using a linear approach. Based on this assumption, at each step, the input is moved in a *greedy* way towards the (estimated) closest boundary, using the perturbation r_i (see equations (3) and (4)). However, there could exist alternative perturbations that, with the same amount of distortion employed by DeepFool, can reach a different decision region, or even reach it with less distortion. Being able to find such *shortcuts* in an efficient way can increase the effectiveness of the attack, or reduce the required amount of perturbation to fool the model. Note that, if this is possible, it is because the linear approximation of the decision regions is not sufficiently accurate, and, therefore, we refer to them as *gaps* or *holes* in the algorithm. Based on this hypothesis, in this section we propose two different strategies to explore, during the original attack process, the existence of such gaps in the algorithm.

Let x_i be the input sample at step i, assuming that $f(x_i) = f(x_0) = y_c$. Let $\widehat{W} = \{\hat{w}_1, \dots \hat{w}_{c-1}, \hat{w}_{c+1} \dots, \hat{w}_k\}$ be the set of normalized direction vectors computed by DeepFool for each class (except y_c), where:

$$\hat{w}_j = \frac{w'_j}{||w'_j||_2}. \tag{8}$$

It is worth mentioning that, following the criterion described in equation (3), the direction $\hat{w}_l$ is the one that will be followed by DeepFool according to its *greedy* criterion.

The first search strategy that we propose, named *Gap Finder Local Search* (GFLS), consists of exploring all the possible directions in $\widehat{W} - \{\hat{w}_l\}$ at each step, projected in the sphere of radius $||r_i||_2$ and centered at the current point x_i. That is, we consider a *gap* if, with a perturbation amount of $||r_i||_2$, we can change the output of the model by pushing x_i in any other direction $\hat{w}_j \in \widehat{W} - \{\hat{w}_l\}$:

$$\exists j \neq l : f\left(x_i + \hat{w}_j \cdot ||r_i||_2\right) \neq f(x_i). \tag{9}$$

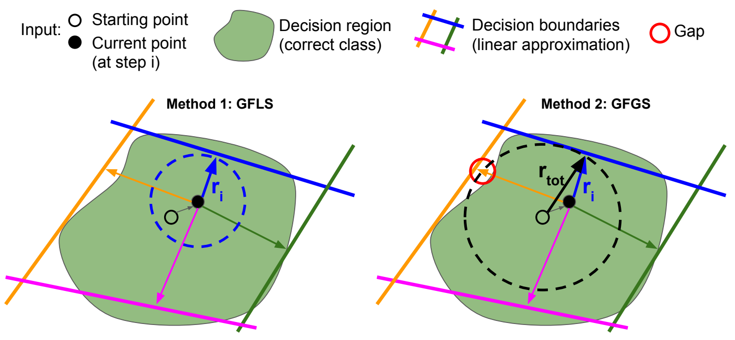
Input:
Starting point
Current point (at step i)
Decision region (correct class)
Decision boundaries (linear approximation)
Gap
Method 1: GFLS
Method 2: GFGS
r_i
r_{tot}

Figure 1: Illustration of the two search methodologies employed to find gaps: *Gap Finder Local Search* (GFLS) and *Gap Finder Global Search* (GFGS).

In the second strategy, instead of taking the *local* neighborhood of x_i as reference, we will take as reference the initial point x_0, and consider the accumulated perturbation $r_{tot} = (x_i - x_0) + r_i$ at the end of the step i. We will refer to this second strategy as *Gap Finder Global Search* (GFGS). This criterion is more suitable if we want to check alternative directions, while ensuring that the *total* perturbation amount that those directions produce is the same as the one produced by DeepFool. Thus, now the search space will be bounded to the surface of the sphere of radius $||r_{tot}||_2$ and centered at the starting point x_0:

$$\exists j \neq l : f\left(x_0 + r^j_{tot} \cdot \frac{||r_{tot}||_2}{||r^j_{tot}||_2}\right) \neq f(x_i)\ ,\ r^j_{tot} = (x_i - x_0) + \frac{|f'_j|}{||w'_j||^2_2} w'_j. \quad (10)$$

An illustration of the two search strategies is provided in Figure 1.

Note that, in both cases, we assume that a gap is found if an alternative perturbation (that is, different to the one proposed by DeepFool) is capable of changing the output of the model, with no additional regard for the incorrect class that is produced. Thus, the proposed methodologies are particularly well suited for untargeted attacks, as in targeted attacks we focus on reaching a particular output rather than producing any possible incorrect class.

5 Evaluating Gaps in the Linear Approximation of the Decision Boundary

5.1 Case of Study

We will use the Speech Command Dataset [28], which consists of a set of 16-bit WAV audio files of 30 different spoken commands. The dataset can be downloaded from [1]. The duration of all the files is the same in all the audio

[1] http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz

clips, exactly 1 second. The sample-rate of the audio signals is 16kHz, so every audio signal is composed of 16000 values.

As in previous publications [1, 28], we focused on the following set of classes out of the 30 different classes in the original dataset: *Yes, No, Up, Down, Left, Right, On, Off, Stop*, and *Go*. In addition to this set, we will consider two special classes for a more realistic setup: *Unknown* and *Silence*. We used a Convolutional Neural Network as the classifiation model, based on the architecture proposed in [20], which is considered a benchmark in speech command classification [1, 4, 28, 32].

To explore the existence of gaps, we selected a set of $N = 500$ files per class from the training set, forming a total of 6000 inputs. For each audio, a default untargeted DeepFool attack is executed, and at each step of the process the two strategies proposed in Section 4 are applied to search for gaps in the vicinity of the inputs. We launched the experiment for the following maximum distortion thresholds, with which the ℓ_2 norm of the perturbations will be bounded: $\varepsilon = \{0.05, 0.1, 0.15\}$. Our implementation is available upon request.

5.2 Analysis of the Results

In this section we evaluate with which frequency *gaps* were found in the linear approximation of the decision boundary employed by DeepFool, as described in Section 4. We also analayze the gain in terms of effectiveness with respect to the original algorithm, or the insights that the study of these gaps provide regarding the structure of the decision space of the classifier.

First, we report the percentage of inputs for which, at any step of the algorithm, the introduced search methods found a *gap*. Table 1 contains the obtained percentages for different distortion thresholds. As can be seen, for the majority of the classes, the GFLS approach was capable of finding a gap for approximately 30% of the inputs samples. With the GFGS approach, however, we only found gaps for less than 10% of the inputs for most of the classes. This can reflect that the accumulated perturbation, after multiple steps, achieves a much better approximation of the optimal perturbation r^* than the individual perturbations applied at each step.

In spite of these positive results, if we analyze the iteration number in which those gaps were found, it can be seen that a very high percentage of gaps was found in the last iteration of the process, considering the last step the one in which $f(x_{i+1}) \neq f(x_i) = f(x_0)$ is satisfied. This information is displayed in Table 2. These percentages are particularly high for the GFLS approach, in which more than approximately 80% of the gaps were found in the last step, independently of the class. Although this is also true for some classes for the GFGS method, the percentages are considerably lower for the rest, around 40% in the lowest cases.

These results suggest, especially for the local strategy, that the gaps are mainly found in the proximity of the decision boundary. Therefore, we can hypothesize that, even through gaps, the input is being pushed towards the same decision boundary that is reached using the default untargeted attack. This hypothesis

Table 1: Percentage of inputs for which gaps were found in the linear approximation of the decision boundaries, during the attack process of DeepFool.

Class	GFLS ε			GFGS ε		
	0.05	0.10	0.15	0.05	0.10	0.15
Go	31.00	31.20	31.20	9.20	9.20	9.20
Stop	26.60	28.80	29.80	6.20	7.40	8.60
Off	30.80	31.40	31.40	21.40	22.00	22.20
On	31.80	32.00	32.00	7.40	7.80	7.80
Right	21.80	22.20	22.40	3.40	3.40	3.40
Left	27.60	28.60	28.80	5.60	6.00	6.40
Down	25.20	26.80	27.00	7.20	8.00	8.00
Up	29.80	30.40	30.60	7.00	7.40	7.40
No	27.20	27.60	27.60	6.40	6.60	6.80
Yes	26.80	28.00	28.40	11.60	12.60	14.20
Unknown	27.00	27.20	27.20	6.40	6.60	6.60
Silence	6.00	7.60	8.20	22.40	31.20	41.20

would also explain why the local search followed in GFLS achieved considerably more gaps than GFGS, as in the close proximity of the decision boundary there is a higher chance of surpassing the boundary by moving the sample to other directions. To validate this hypothesis, we computed the percentage of gaps that reach a different class than the default untargeted DeepFool algorithm. According to the results, for both methods, only for a percentage below 1% of the inputs is it possible to produce a wrong output class different to that which is produced by the default attack.

All these results might also suggest that the linear approximation of the decision boundaries employed by DeepFool is highly accurate in our case, and as a consequence, that the generated perturbations are close to the optimal ones. However, another explanation can be that, in the proximity of the natural input samples, there is generally a decision boundary for one class much closer than those corresponding to the rest of the classes. In fact, if we compute the frequency with which each input is classified as another class after being fooled by the original DeepFool algorithm, we can see that, for the majority of the classes, there are always one or two classes with higher frequency. This information is shown in Figure 2, for the case of $\varepsilon = 0.15$, although the same pattern is given for the other distortion thresholds tried.

To continue with the analysis, in order to assess the distortion gain that the gaps suppose, if any, we also compared the distortion introduced by the gaps with the one introduced by the untargeted attacks. The comparison is made by considering those inputs for which a perturbation capable of fooling the model was found with both attack types. If more than one possible gap was found for one input, the one with the lowest perturbation has been considered. The results are shown in Figure 3, for the different distortion thresholds.[2] As can be seen, the gain is negligible, showing again that the found gaps are not

[2]The outliers have been removed for the sake of visualization.

Table 2: Percentage of gaps that were found in the last step of the process. We consider the last step the one in which, when we apply the last perturbation to the (still correctly classified) input, we reach a new decision region, that is, when $f(x_{i+1}) \neq f(x_i) = f(x_0)$.

Class	GFLS			GFGS		
	ε			ε		
	0.05	0.10	0.15	0.05	0.10	0.15
Go	80.65	80.77	80.77	50.00	50.00	50.00
Stop	87.97	88.89	87.92	67.74	72.97	76.74
Off	83.77	84.08	84.08	65.14	65.18	65.49
On	86.16	86.25	86.25	62.16	64.10	64.10
Right	89.91	90.09	90.18	82.35	82.35	82.35
Left	78.99	79.72	79.17	39.29	40.00	43.75
Down	83.33	83.58	83.70	55.56	55.00	55.00
Up	87.92	87.50	87.58	74.29	75.68	75.68
No	80.88	81.16	81.16	42.42	41.18	42.86
Yes	91.11	90.78	90.91	65.52	61.90	63.89
Unknown	81.48	81.62	81.62	40.62	42.42	42.42
Silence	80.00	78.95	76.19	67.26	71.07	73.93

very different to the perturbation provided by DeepFool, which reinforces our previous hypothesis.

Finally, we also tested the hypothesis that, through the gaps, it could be possible to generate perturbations capable of fooling the model for those inputs for which, through the original algorithm, it would not be possible to find such a perturbation. However, according to the results obtained by the GFLS strategy, in no case we were able to fool the model using only the perturbations generated through gaps. For the GFGS strategy, the percentage of inputs for which this happens is below 1%. Thus, for almost all the inputs for which gaps were found, we could also find a perturbation capable of fooling the model using the default algorithm.

Similarly, if we compare the results with the targeted version of DeepFool, relying on gaps to find targeted attacks is much less effective than generating them by moving the input directly towards the decision boundary of the target class. Only in less than 1% of the cases were gaps capable of reaching a particular class that could not be reached through the targeted attack, according to both search strategies. Moreover, the difference in terms of distortion level is negligible in this case also, as is shown in Figure 4.

6 Conclusion

In this paper, we have proposed two different strategies to search for inaccuracies in the linear approximation of the decision boundaries employed by the DeepFool algorithm, a state-of-the-art adversarial attack, to assess whether they can increase the effectiveness of the attack, as well as a framework to study the properties of the decision spaces of machine learning classifiers. We

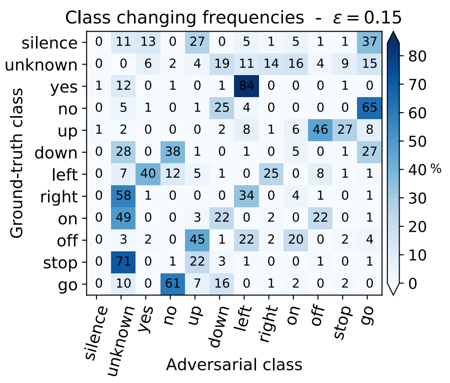
Figure 2: Class changing frequencies caused by the default untargeted DeepFool algorithm. Each row in the matrix represents the percentage of cases changed from the corresponding ground-truth class to a different class.

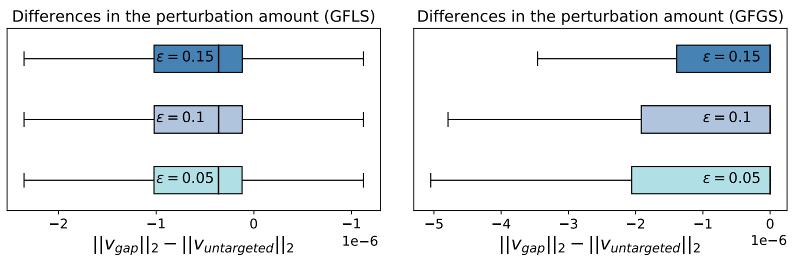
Figure 3: Comparison of the ℓ_2 norm between the perturbations obtained using gaps, v_{gap}, and the perturbation obtained with the default untargeted attack, $v_{untargeted}$.

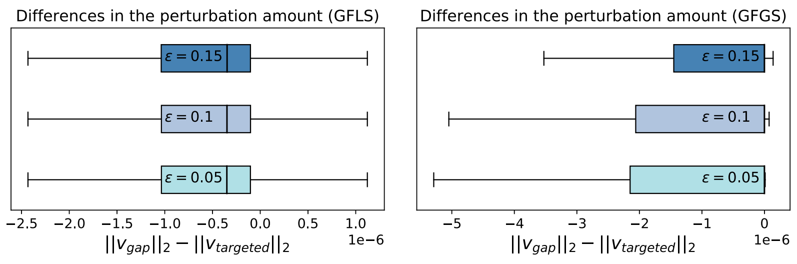
Figure 4: Comparison of the ℓ_2 norm between the perturbations obtained using gaps, v_{gap}, and the perturbation obtained with the default targeted attack, $v_{targeted}$.

experimentally tested our approach for a Convolutional Neural Network in the speech command classification task, an exemplary machine learning model in the audio domain.

The results obtained revealed that, in approximately 30% of the inputs evaluated, the approach introduced was capable of exploiting those inaccuracies to find alternative directions in which the model can be fooled, with a negligible overhead with respect to the original algorithm, although the gain in terms of effectiveness was not significant enough to consider it an improvement. In fact, we discovered that most of these gaps lead to perturbations highly similar to those found by the default algorithm. We intend to improve the results by extending the search strategies in future work.

As future research lines, we also believe that the comparison of the geometrical properties of the decision boundaries of different machine learning models can provide insightful contributions regarding their vulnerability to adversarial attacks. For this reason, we intend to extend our methodology to evaluate a wider range of decision models, including more complex structures, such as recurrent neural networks, as well as to other tasks and domains, for instance, speech transcription, computer vision tasks or natural language processing problems.

7 Acknowledgments

This work is supported by the Basque Government (BERC 2018-2021 and ELKARTEK programs, IT1244-19, and PRE_2019_1_0128 predoctoral grant), by the Spanish Ministry of Economy and Competitiveness MINECO (projects TIN2016-78365-R and PID2019-104966GB-I00) and by the Spanish Ministry of Science, Innovation and Universities (FPU19/03231 predoctoral grant). Jose A. Lozano acknowledges support by the Spanish Ministry of Science, Innovation and Universities through BCAM Severo Ochoa accreditation (SEV-2017-0718).

References

[1] Alzantot, M., Balaji, B., Srivastava, M.: Did you hear that? Adversarial examples against automatic speech recognition. arXiv preprint arXiv:1801.00554 (2018)

[2] Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 39–57. IEEE (2017)

[3] Cisse, M., Adi, Y., Neverova, N., Keshet, J.: Houdini: Fooling deep structured prediction models. arXiv preprint arXiv:1707.05373 (2017)

[4] Du, T., Ji, S., Li, J., Gu, Q., Wang, T., Beyah, R.: Sirenattack: Generating adversarial audio for end-to-end acoustic systems. arXiv preprint arXiv:1901.07846 (2019)

[5] Esmaeilpour, M., Cardinal, P., Lameiras Koerich, A.: A robust approach for securing audio classification against adversarial attacks. IEEE Transactions on Information Forensics and Security **15**, 2147–2159 (2020)

[6] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)

[7] Grosse, K., Manoharan, P., Papernot, N., Backes, M., McDaniel, P.: On the (statistical) detection of adversarial examples. arXiv preprint arXiv:1702.06280 (2017)

[8] Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., Madry, A.: Adversarial examples are not bugs, they are features. arXiv preprint arXiv:1905.02175 (2019)

[9] Karimi, H., Tang, J.: Decision boundary of deep neural networks: Challenges and opportunities. In: Proceedings of the 13th International Conference on Web Search and Data Mining. pp. 919–920 (2020)

[10] Khoury, M., Hadfield-Menell, D.: On the geometry of adversarial examples. arXiv preprint arXiv:1811.00525 (2018)

[11] Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 (2016)

[12] Luo, B., Liu, Y., Wei, L., Xu, Q.: Towards imperceptible and robust adversarial example attacks against neural networks. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)

[13] Mickisch, D., Assion, F., Greßner, F., Günther, W., Motta, M.: Understanding the decision boundary of deep neural networks: An empirical study. arXiv preprint arXiv:2002.01810 (2020)

[14] Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1765–1773 (2017)

[15] Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P., Soatto, S.: Analysis of universal adversarial perturbations. arXiv preprint arXiv:1705.09554 (2017)

[16] Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2574–2582 (2016)

[17] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 372–387. IEEE (2016)

[18] Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE Symposium on Security and Privacy (SP). pp. 582–597. IEEE (2016)

[19] Roos, M.J.: Utilizing a null class to restrict decision spaces and defend against neural network adversarial attacks. arXiv preprint arXiv:2002.10084 (2020)

[20] Sainath, T.N., Parada, C.: Convolutional neural networks for small-footprint keyword spotting. In: Sixteenth Annual Conference of the International Speech Communication Association. pp. 1478–1482 (2015)

[21] Samangouei, P., Kabkab, M., Chellappa, R.: Defense-GAN: Protecting classifiers against adversarial attacks using generative models. arXiv preprint arXiv:1805.06605 (2018)

[22] Stutz, D., Hein, M., Schiele, B.: Disentangling adversarial robustness and generalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6976–6987 (2019)

[23] Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation **23**(5), 828–841 (2019)

[24] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: 2nd International Conference on Learning Representations (2014)

[25] Tanay, T., Griffin, L.: A boundary tilting persepective on the phenomenon of adversarial examples. arXiv preprint arXiv:1608.07690 (2016)

[26] Tramèr, F., Papernot, N., Goodfellow, I., Boneh, D., McDaniel, P.: The space of transferable adversarial examples. arXiv preprint arXiv:1704.03453 (2017)

[27] Vadillo, J., Santana, R.: Universal adversarial examples in speech command classification. arXiv preprint arXiv:1911.10182 (2019)

[28] Warden, P.: Speech commands: A dataset for limited-vocabulary speech recognition. arXiv preprint arXiv:1804.03209 (2018)

[29] Yang, Z., Li, B., Chen, P.Y., Song, D.: Characterizing audio adversarial examples using temporal dependency. arXiv preprint arXiv:1809.10875 (2018)

[30] Yousefzadeh, R., O'Leary, D.P.: Investigating decision boundaries of trained neural networks. arXiv preprint arXiv:1908.02802 (2019)

[31] Yuan, X., He, P., Zhu, Q., Li, X.: Adversarial examples: Attacks and defenses for deep learning. IEEE transactions on neural networks and learning systems **30**(9), 2805–2824 (2019)

[32] Zhang, Y., Suda, N., Lai, L., Chandra, V.: Hello edge: Keyword spotting on microcontrollers. arXiv preprint arXiv:1711.07128 (2017)