U-DEEPDIG: SCALABLE DEEP DECISION BOUNDARY INSTANCE GENERATION

Jane Berk[⋆]*, Martin Jaszewski*[⋆]*, Charles-Alban Deledalle*[†]*, and Shibin Parameswaran*[⋆]

[⋆]Naval Information Warfare Center Pacific, 53560 Hull Street, San Diego, CA
[†]Brain Corp, 10182 Telesis Ct, San Diego, CA

ABSTRACT

For more than a decade, deep learning algorithms have consistently achieved and improved upon the state-of-the-art performance on image classification tasks. However, there is a general lack of understanding and knowledge about the decision boundaries carved by these modern deep neural network architectures. Recently, an algorithm called DeepDIG was introduced to generate boundary instances between two classes based on the decision regions defined by any deep neural network classifier. Although it is very effective in generating boundary instances, the underlying algorithm was designed to work with two classes at a time in a non-commutative fashion which makes it ill-suited for multi-class problems with hundreds of classes. In this work, we extend the DeepDIG algorithm so that it scales linearly with the number of classes. We show that the proposed U-DeepDIG algorithm maintains the efficacy of the original DeepDIG algorithm while being scalable and more efficient when applied to larger classification problems. We demonstrate this by applying our algorithm on MNIST, Fashion-MNIST and CIFAR10 datasets. In addition to qualitative comparisons, we also perform extensive quantitative comparison by analyzing the margin between the class boundaries and the instances generated.

1. INTRODUCTION

Deep learning algorithms have become the main workhorse of many supervised learning tasks such as image classification [1, 2, 3]. With its amenability to transfer learning [4] and thanks to highly optimized user-friendly frameworks, even a beginner in machine learning can are readily train a deep learning model for their task and achieve impressive results, given sufficient annotated training data and computing resources (e.g. GPUs).

Even though the barrier to adopt deep learning algorithms has dropped in the last decade, there is a lack of understanding about the decision boundaries defined by these algorithms. The extremely high dimensional input spaces and the complexity of a modern deep learning algorithm with millions of parameters make exploring the characteristics of decision boundaries a challenging task. Recent works have motivated the study and analysis of decision boundaries through the lens of robustness [5, 6, 7], generalizability [8], margin dynamics during training [9, 10], or through generating instances close to decision regions [11, 12, 13]. One such approach that falls into the last category, called DeepDIG [11, 12], utilizes autoencoder-based adversarial example generation to create instances close to a given model's decision boundary, irrespective of model depth or complexity.

Although DeepDIG is effective in generating boundary instances, it has two major drawbacks namely the underlying algorithm can only consider two classes at a time (binary) and is non-commutative. This makes DeepDIG ill-suited for models with hundreds of classes. In this study, we introduce a straightforward yet innovative extension to DeepDIG so that the number of boundaries to consider scales linearly with number of classes, rather than combinatorially (as is the case with DeepDIG). The proposed approach, U-DeepDIG, is as effective in generating boundary instances as its original binary version while being scalable to larger problems. We ascertain the efficacy of the proposed algorithm by applying this on three benchmark datasets.

2. BACKGROUND

In this section we describe the Deep Decision boundary Instance Generation (DeepDIG) algorithm [11, 12] in detail and provide a brief summary of other related works.

2.1. DeepDIG

Given a pre-trained model and a particular dataset, DeepDIG is optimized to generate borderline instances near the decision boundary defined by the model between *two* classes, where the classification probabilities are as close as possible. DeepDIG consists of three main components. The first component (I) utilizes an autoencoder (AE) to generate *targeted* adversarial examples from a source class, s, to a target class, t. In the second component (II) another AE is used to adversarially attack the newly created adversarial example generated by Component I (resulting in the sample moving back to class s). The final component (III) takes these two adversarial examples, and uses a binary search algorithm to search on the

[⋆]The authors would like to thank the NIWC Pacific Naval Innovative Science and Engineering (NISE) program for the support.

line connecting outputs of components I and II to find refined instances that are sufficiently close to the decision boundary. In short, DeepDIG generates two versions of a datapoint perturbed to fall on either side of the boundary and finds the inflection point in decision region along the line connecting them.

The loss functions for the AEs of Component I and II, A_I and A_{II}, are shown in Equations 1 and 2 below:

$$\mathcal{L}_I(A_I) = \sum_{\forall x_s} \left[||x_s - \hat{x}_t||_2^2 + \alpha \cdot CE\left(f(\hat{x}_t), \vec{t}\right) \right] \quad (1)$$

$$\mathcal{L}_{II}(A_{II}) = \sum_{\forall \hat{x}_t} \left[||\hat{x}_t - \hat{x}_s||_2^2 + \alpha \cdot CE\left(f(\hat{x}_s), \vec{s}\right) \right] \quad (2)$$

In the above loss functions, x_s is the original sample of class s, $\hat{x}_t = A_I(x_s)$ is the adversarial sample generated by AE of Component I that is classified into class t, $\hat{x}_s = A_{II}(x_t)$ is the adversarial sample generated by AE of Component II classified as s, $f(\cdot)$ is the classifier whose i-th element is the softmax output such that $f_i(x) = P[x \in C_i]$ (the probability that x belongs to class i), $\vec{k}$ is a c-dimensional one-hot embedded vector where the k-th entry is 1 and the rest is equal to 0, and α controls the trade-off between the reconstruction error and the adversarial classification constraint measured by cross-entropy loss.

The results of Component I and II allow for two instances to be created near the decision boundary, but more importantly, on either side of it. As ablated by Karimi et al. [12], this increases the success and decreases the time needed for Component III to complete binary search algorithm to identify a refined boundary point, x_b, along the line connecting these two samples ($\hat{x}_t$ and $\hat{x}_s$), with the probability of it belonging to classes t and s are almost equal (within ϵ), i.e., $P(x_b \in C_s) \approx P(x_b \in C_t)$.

While the DeepDIG algorithm may be an efficient method for finding the decision boundary, the foundational methodology of the AEs is targeted adversarial attacks. This approach forces a multi-class classification to be viewed as multiple binary classification problems that are non-commutative. For instance, consider the following 4 class problem and the boundaries defined by a classifier in Figure 1. Under the DeepDIG framework, we have to consider $2 \times ({}_4C_2) = 12$ cases to fully characterize the decision regions of that model. We have illustrated the 12 cases in the 6 boxes in the top diagram of Figure 1, each showing the non-commutative source-target pairings (e.g. red→green and green→red). The parts of the decision boundary that stays the same for the chosen 2 classes is shown in black. The part that changes depending on the source class and is shown in the color of the source class datapoints. Note that, in this setting, even if we are only interested in the boundary of one particular class, all 12 cases will have to be considered since the classes sharing boundaries will not be known *a priori*.

Consider the datasets ImageNet [1] with 1000 classes or CIFAR100 [14] with 100 classes, the decision boundaries would need to be calculated 999,000 times, and 9,900 times respectively; the computational needs required for these calculations is highly impractical.

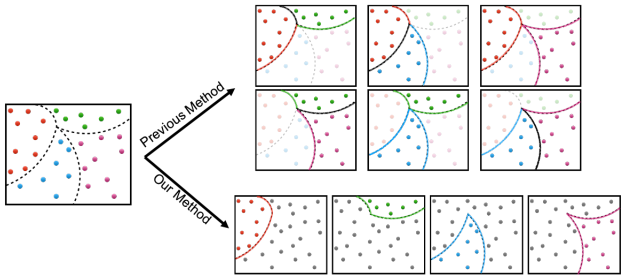


Fig. 1: Decision boundary analysis using Targeted (top) versus Untargeted DeepDIG (bottom).

2.2. Other related work

Approaches adopted by recent works that study decision boundaries in their input space vary greatly from empirical observations of decision margins [10] and Brownian motions processes [6], to topology analysis [15]. Due to the focus of this work, we list methods that use instance generation to understand model boundary characteristics. Boundary instances can be used for adversarial training to create more robust classifiers, for characterizing the curvature of the decision boundaries, and for studying the dynamics of boundary and class neighborhoods during model training. He et al. [13] searches for boundaries by sampling perturbations in random directions and looking for predictions changes along them. This approach gets computationally prohibitive to cover the input space satisfactorily as dimensionality increases. Others have [8, 15] searched for the inflection point (or flip points) along a line connecting points in two classes. Fawzi et al. [7] uses a similar method of connecting two data points and analyzing the decision regions passed through as one travels from one point to the other. Another approach is to search over a line connecting a point to its adversarial copy that has (almost) equal prediction probability for either classes [16]. Karimi et al. [12] demonstrated that similar approaches tend to generate non-natural images or have lower success rates of finding boundary points.

3. UNTARGETED DEEPDIG ALGORITHM

To alleviate the issues presented by the targeted DeepDIG algorithm, which we will call T-DeepDIG, we propose to extend it by switching the targeted adversarial loss by an untargeted loss, U-DeepDIG. To that end, we want our approach to find the decision boundary between the chosen source class and all other classes, thus scaling the problem linearly with the number of classes.

For instance, again consider the four class problem described in Figure 1, in the untargeted case, there are only

4 boundaries that need to be determined; the bottom set of boxes in Figure 1 shows this case. With an untargeted approach, there is no ambiguity in the decision boundary since the union of the source and target class datapoints covers the entire dataset; i.e. each image in Figure 1 depicts a single case.

3.1. Untargeted Adversarial Loss

Consider the second term of the loss function for Component I, shown in Equation 1, and the variable representing the c-dimensional one-hot vector. For the targeted case, the adversarial cross entropy loss is given by,

$$\begin{aligned} CE(f(x), \vec{t}) &= -\sum_{i=1}^{c} t_i \log(P[x \in C_i]) \\ &= -\log[f_t(x)] \end{aligned} \tag{3}$$

where t_i is the i-th element of the one-hot embedded vector and $P(x \in C_i)$ is the probability that x belongs to class i. Since only the t-th element of $\vec{t}$ is non-zero and equal to 1, the only term that contributes to loss is $P[x \in C_t] = f_t(x)$, i.e., a loss is incurred if x is not classified as belonging to class t. (Note that this is the same as regular classification cross-entropy loss with the correct class switched out with targeted class). In order to create an untargeted version of DeepDIG, this is the term that needs to be modified.

To achieve a cost that will force the classifier to choose *any* class other than source (s), we have to make a minor modification to the cross-entropy loss.

$$\begin{aligned} uCE(f(x), \vec{s}) &= -\sum_{i=1}^{c} s_i \log(P[x \notin C_i]) \\ &= -\log[1 - f_s(x)], \end{aligned} \tag{4}$$

This loss encourages x to be classified as any class other than the correct one s. Hence, by replacing the $CE(\cdot)$ term in Eq. 1 with $uCE(\cdot)$ shown in Eq. 4, the AE will be forced to learn a mapping such that the reconstructed sample is classified as belonging to a class that is not the correct source class s; thus producing an untargeted adversarial loss. The loss function for Component II, Equation 2 will remain unchanged because the goal of this component is to generate an example that will be classified as belonging to the source class; hence, the targeted loss is appropriate for this component. Similarly, the algorithm for Component III will remain unchanged as well.

4. EXPERIMENTAL SETUP

To evaluate the boundary instance generation capability of the proposed method, U-DeepDIG, compared to the targeted version, T-DeepDig [11, 12], we apply the algorithms to MNIST [18], Fashion-MNIST [19] and CIFAR10 [14] datasets. In order to make the performance comparisons straightforward, we chose to keep the architectures of the AEs used in Components I and II identical to those used by Karimi et al. [12]. Similarly, we chose architectures used by Karimi et al. [12] to train our classifiers for each dataset, and which then serve as the pre-trained models, i.e. $f(\cdot)$, for testing boundary instance generation. Specifically, to classify MNIST and Fashion-MNIST we use a simple Convolutional Neural Networks (CNN) and for CIFAR10 we use ResNet [2]. The classifiers were trained to reach competitive performance levels comparable to those presented in Karimi et al. [12, 11]. The classifiers used per dataset and their respective classification performance on the respective *standard* test sets are shown in Figure 3(a). Please note that architectures and other model parameters are kept constant between U-DeepDIG and T-DeepDIG in our experiments. Hence, they do not affect the evaluations and comparisons presented in the next section. We kindly refer interested readers to [12] for more details on other training details.

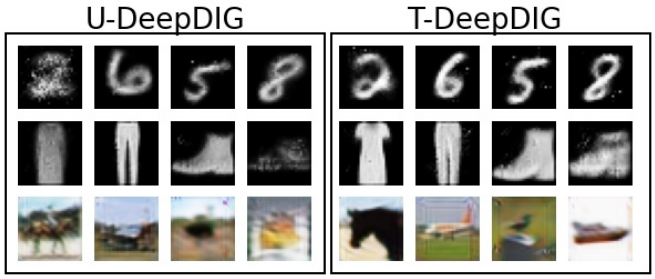


Fig. 2: U-DeepDIG (ours) and T-DeepDIG boundary samples

	MNIST	Fashion-MNIST	CIFAR10
U-DeepDIG (ours)	98.86%	89.09%	98.49%
T-DeepDIG[1]	86.79%	86.26%	89.71%
T-DeepDIG[2]	98.66%	98.93%	70.60%

[1] Results generated by us using Karimi et al.'s implementation [17]
[2] Results copied from Karimi et al. [12]

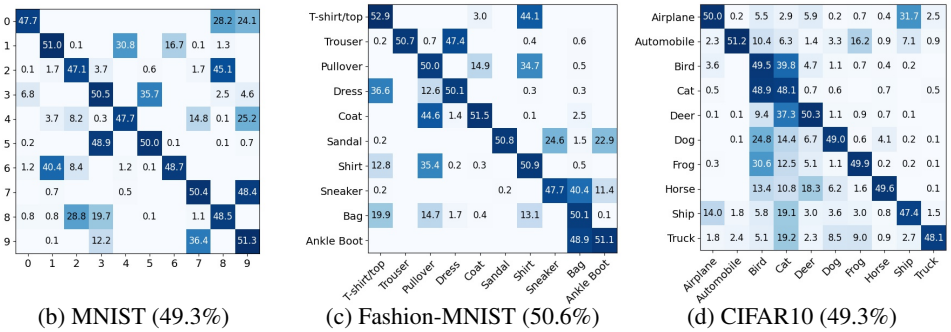
Table 1: Comparison of decision boundary sample generation *success rates* of U-DeepDIG (ours) and T-DeepDIG [11, 12].

5. RESULTS

In this section, we present the qualitative and quantitative evaluation of the ability of U-DeepDIG algorithm to generate boundary instances. Primarily, we compared the performance of our approach to the targeted DeepDIG algorithm (T-DeepDIG). Applying U-DeepDIG to the 10-class problems of MNIST, Fashion-MNIST and CIFAR10 require one run per class resulting in 10 runs per dataset. However, the T-DeepDIG algorithm, due to the targeted approach, results in 90 runs per dataset. Even with this immense efficiency improvement, U-DeepDIG manages to create boundary in-

Dataset	Arch.	Acc.
MNIST	CNN	98.87%
Fashion MNIST	CNN	89.09%
CIFAR10	ResNet	82.84%

(a) Model architectures used and the accuracies achieved on **standard** test sets



(b) MNIST (49.3%) (c) Fashion-MNIST (50.6%) (d) CIFAR10 (49.3%)

Fig. 3: Classifier performance on generated border instances. The accuracy on the generated instances is shown in parentheses.

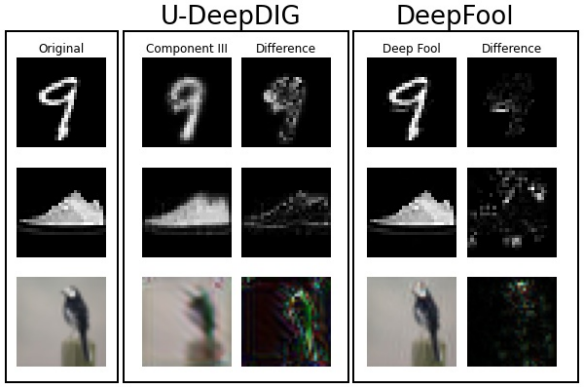


Fig. 4: Sample border instances generated by U-DeepDIG and Deepfool for MNIST, Fashion-MNIST and CIFAR10. Perturbations (differences) are scaled so that max intensity is equal to 1.

stances that are qualitatively comparable and as realistic as its targeted counterpart (T-DeepDIG). Figure 2 shows some of the examples generated by U-DeepDIG and T-DeepDIG.

One of the metrics used by Karimi et al. [12] to measure T-DeepDIG's efficacy is *success rate*; i.e. the ratio of input data samples to the number of successful borderline instances generated. A comparison of success rates of U-DeepDIG and T-DeepDIG are shown in Table 1. For Fashion-MNIST and CIFAR10, our attempts to reproduce the baseline (denoted as T-DeepDIG[1]) resulted in significantly different results than those presented in [12] (denoted as T-DeepDIG[2]; we have included both in Table 1. The performance of our approach, U-DeepDIG, is comparable and is often better than the baseline. This is intuitive as finding a boundary with *any* class (untargeted) is easier than with a *particular* class (targeted).

Additionally, we show the confusion matrices in Figures 3b-3d denoting an accuracy drop across all datasets to ∼50% – indicating model uncertainty when applied to samples generated by U-DeepDIG and hence their proximity to a decision boundary. Another advantage of the U-DeepDIG algorithm is that the confusion matrices readily show the classes that share boundary with a given source class and thus cause the most uncertainty, given the current state of the classification model (e.g. Class 0 → Classes {8, 9} in Fig. 3b). This phenomenon is hard to emulate and observe in the T-DeepDIG setting[1].

Finally, we used the Deepfool [20] adversarial attack as a way to measure proximity of generated samples to the decision boundary. Deepfool is designed to find the smallest perturbation needed to make a sample reach its closest boundary. Therefore, this perturbation can provide an approximation of the distance of a sample to the boundary. The average L_2 norms of the Deepfool perturbations of the standard test samples of MNIST, Fashion-MNIST and CIFAR10 are 1.45, 0.19 and 0.51, respectively. These were reduced to 1.47 $\times 10^{-4}$, 7.17 $\times 10^{-4}$ and 1.51 $\times 10^{-4}$, respectively, for borderline instances generated by U-DeepDIG.

Note that Deepfool adverarial attack is also generating boundary instances. Figure 4 compares boundary samples generated by U-DeepDIG, Deepfool outputs, and their corresponding perturbations. Due to the use of AEs, U-DeepDIG perturbations are more structured and concentrated on the foreground than Deepfool perturbations. Furthermore, once trained, the AE in Component I (and II) can generate adversarial samples using unseen test data; whereas, Deepfool has to be run on every sample (but needs no prior training).

6. CONCLUSION

We introduced an efficient and effective algorithm to generate boundary instances of a given pre-trained classifier. The proposed U-DeepDIG approach improves upon the baseline, T-DeepDIG, by making the underlying algorithm scale linearly with number of classes rather than combinatorially. We showed that the resulting approach is as effective as the baseline in creating realistic boundary samples. The boundary proximity of the generated data was validated using Deepfool and drop in classifier performance (uncertainty). The classifier confusions induced by the samples from U-DeepDIG gives indications of the decision neighborhoods, given the current state of the classifier. Studying the dynamics of these confusions during the course of model training is an interesting direction of study for future work.

[1]This may be possible by comparing success rates but unclear if that will be indicative enough since the class confusions are forced in the targeted case

7. REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[4] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*. Springer, 2018, pp. 270–279.

[5] S.-M. Moosavi-Dezfooli, A. Fawzi, J. Uesato, and P. Frossard, "Robustness via curvature regularization, and vice versa," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9078–9086.

[6] B. Georgiev, L. Franken, and M. Mukherjee, "Heating up decision boundaries: isocapacitory saturation, adversarial scenarios and generalization bounds," *arXiv preprint arXiv:2101.06061*, 2021.

[7] A. Fawzi, S.-M. Moosavi-Dezfooli, P. Frossard, and S. Soatto, "Empirical study of the topology and geometry of deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3762–3770.

[8] S. Guan and M. Loew, "Analysis of generalizability of deep neural networks based on the complexity of decision boundary," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2020, pp. 101–106.

[9] M. Eslami, H. Eramian, M. Gameiro, W. Kalies, and K. Mischaikow, "Extracting global dynamics of loss landscape in deep learning models," *arXiv preprint arXiv:2106.07683*, 2021.

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

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

[12] H. Karimi, T. Derr, and J. Tang, "Characterizing the decision boundary of deep neural networks," *arXiv preprint arXiv:1912.11460*, 2019.

[13] W. He, B. Li, and D. Song, "Decision boundary analysis of adversarial examples," in *International Conference on Learning Representations*, 2018.

[14] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[15] R. Yousefzadeh and D. P. O'Leary, "Investigating decision boundaries of trained neural networks," *arXiv preprint arXiv:1908.02802*, 2019.

[16] S. Rezaei and X. Liu, "On the difficulty of membership inference attacks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7892–7900.

[17] H. Karimi, "Deepdig," https://github.com/hamidkarimi/DeepDIG.git, 2019.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[19] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[20] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.