MUTEN: Mutant-Based Ensembles for Boosting Gradient-Based Adversarial Attack

Qiang Hu*, Yuejun Guo†, Maxime Cordy*, Mike Papadakis* and Yves Le Traon*
*University of Luxembourg, Luxembourg
†Luxembourg Institute of Science and Technology, Luxembourg

***Abstract*—Mutation testing (MT) for deep learning (DL) has gained huge attention in the past few years. However, how MT can really help DL is still unclear. In this paper, we introduce one promising direction for the usage of mutants. Specifically, since mutants can be seen as one kind of ensemble model and ensemble model can be used to boost the adversarial attack, we propose MUTEN, which applies the attack on mutants to improve the success rate of well-known attacks against gradient-masking models. Experimental results on MNIST, SVHN, and CIFAR-10 show that MUTEN can increase the success rate of four attacks by up to 45%. Furthermore, experiments on four defense approaches, *bit-depth reduction*, *JPEG compression*, *Defensive distillation*, and *Label smoothing*, demonstrate that MUTEN can break the defense models effectively by enhancing the attacks with the success rate of up to 96%.**

I. Introduction

Recently, researchers [1], [2] tried to employ mutation testing (MT), a famous software testing technique, to help test deep learning models. To that end, multiple mutation operators that are specifically designed for DNNs have been proposed. However, there are always doubts about the meaningfulness of the existing direction of MT for DNNs [3]. Different from the conventional software systems whose test oracle is easy to be defined, DNNs are data-driven and built by complex polynomial iteration. It is hard to analyze the quality of test suites and define the bug of DNNs. How to use MT to ensure the quality of test suites and further reveal the *buggy behavior* of DNNs is unclear. As a result, the real usage of existing proposed mutation operators is unknown. In this paper, we demonstrate that mutants can boost adversarial example generation.

Roughly speaking, an adversarial example is typically crafted by adding a subtle perturbation to a benign input in a way that misleads a DNN model. Adversarial examples are useful not only to reveal security threats in DNNs, but also to improve their robustness, e.g., through adversarial training [4]. Many adversarial attacks have been investigated. In general, there are two types of methods according to accessible information. Black-box attacks have no knowledge of the model, while white-box ones have full access to model information such as its architecture, gradient, and weights. Most white-box adversarial attacks are *gradient-based*, in the sense that they utilize the gradient of the loss function with the hope to compute the perturbation direction that will maximize the likelihood of misclassification. To ensure the secure usage of DNNs and defend against gradient-based attacks, gradient masking [5] has been proposed to obfuscate gradients to reduce the attack success rate of attacks.

We propose MUTEN, a fast and effective way to attack models with masked gradients. The effectiveness of our approach leans on Liu *et al.*'s work [6], where they show that adversarial examples crafted from an ensemble of surrogate models transfer relatively well to the original model. Unlike their method, though, MUTEN avoids the prohibitive cost of training multiple models. More precisely, we generate a set of *mutant* models, obtained by altering the original one (e.g., by introducing random noise into its weights). Working in white-box settings, we also include the original model in the ensemble. Thus, the overhead of attacking the ensemble is limited to applying the attack to each mutant and computing the average of the perturbations.

Given that there exist many ways to mutate a model [1], we utilize mutation combinations to increase attack success rates. Our hypothesis is that an effective ensemble should contain *a diverse set of accurate mutants*. On the one hand, we ensure that the mutants retain the performance (test accuracy) of their original model by controlling the proportion of modified weights, neurons, or layers. On the other hand, we propose a greedy algorithm to select *diverse* mutants where the diversity is measured by the centered kernel alignment (CKA) metric [7] and PageRank algorithm [8]. To summarize, the main contributions of this paper are:

1) We introduce a new direction for the usage of mutation testing on real mML problems and propose MUTEN, a novel approach to boost the effectiveness of gradient-based attacks using mutants by up to 45%.
2) We evaluate MUTEN against four defense approaches and show that MUTEN can effectively break the defenses with a success rate of up to 96%.

II. Background

A. Adversarial Attacks

Adversarial attacks aim at generating adversarial examples that DNNs can not predict correctly by introducing invisible perturbation to existing data samples. From the testing perspective, adversarial examples are usually used as test samples to test the robustness of DNNs. How to effectively generate adversarial examples is an important problem. In this work, we consider boosting gradient-based attacks which are the most effective ones. Two powerful attack methods have been

studied, *Projected Gradient Descent (PGD)* [4] and *Carlini and Wagner attack (C&W)* [9].

B. Mutation of Deep Learning Models

Post-training mutation of DNNs has been applied mainly for quality assurance purposes. Due to the unique characters of DNN models, various mutation operators have been proposed at the source level (modifying the training data or the training program) or at model level [1].

In this paper, we apply model-level mutations, where a mutant is created directly by changing the neurons, weights, or layers slightly without training. In general, modifying the layers requires specific architectures of the DNN models and degrades the performance (accuracy) significantly, and is less applicable. Both the weight- and neuron-level operators work efficiently to generate mutants and are more widely used.

Recent studies have shown the utility of mutation in different tasks. Ma *et al.* [1] propose to mutate test data class by class to figure out the weakness, which is helpful to check for bias in data. Hu *et al.* [2] point out that by a defined killing score metric, the mutants can be used to validate how robust a DNN model is against input data or its segment. In [10], Wang *et al.* assume that the adversarial examples are near the decision boundary, thus, the data that change the labels by different mutants are considered as adversarial examples.

III. Approach

We aim to improve the success rate of gradient-based adversarial attacks applied to gradient-masking models. The main idea of MUTEN is to produce a collection of diverse mutant models to build an ensemble, and attack the ensemble instead of the single original model.

A. Diverse Mutant Generation

Previous research has shown that mutating layers always degrade the performance (test accuracy) significantly [1]. Therefore, we consider 5 operators that only make changes at the weight- and neuron-level. The Gaussian fuzzing (GF) operator adds noise to selected weights. The weight shuffling (WS) rearranges selected weights. The neuron effect blocking (NEB) resets the connection weight of a selected neuron to the next layer to zero. The neuron activation inverse (NAI) operator inverts the activation status of a neuron. The neuron switch (NS) exchanges two neurons within the same layer.

The centered kernel alignment (CKA) [7] metric and the PageRank algorithm [8] are used to control mutant diversity. More precisely, CKA measures the similarity between DNNs. Given the input data X, let H_1 and H_2 be two feature matrices of X by two models, respectively. H_1 and H_2 are considered as the DNN representations. The similarity between H_1 and H_2 is defined by

$$CKA\left(K,L\right)=\frac{\mathrm{HSIC}\left(K,L\right)}{\sqrt{\mathrm{HSIC}\left(K,K\right)\mathrm{HSIC}\left(L,L\right)}} \quad (1)$$

where K and L are the kernel matrices by passing H_1 and H_2 through kernels. HSIC is the Hilbert-Schmidt independence criterion. Like [11], we use the output of the last hidden layer in a DNN as the feature.

The PageRank algorithm aims at measuring the importance/rank of website pages where a page linked to many pages has a high rank. Inspired by this, taking a mutant as a website page and the similarity as the linking weight, we assume that the mutant with a low rank is diverse within the mutant set as it is dissimilar from the others.

Algorithm 1 shows our overall generation method. To increase the diversity of generated mutants, we use 20 pairs of mutation operators and ratios. Note that the mutation ratio R controls the percentage of weights or neurons to be selected in each layer. As applying more mutations has a higher likelihood to decrease model performance, we randomly set it between 1% and 4% following the previous findings of Ma et al. [1]. The mutant set Mu and a similarity matrix D are initialized to be empty, and a counting index $count$ is used to control the termination of the algorithm (Line 1). In each iteration, a pair of mutation operators and ratios is randomly selected from all the candidates (Line 3) to generate a mutant (Line 4). After the first iteration, Mu is updated with a mutant (Lines 5-6). When a new mutant is generated, first, we compute the linear CKA similarity between this mutant and the ones in Mu to update the similarity matrix D (Line 8). If the size of Mu is smaller than n, the procedure continues, otherwise, D is fed into the $pageRank$ function to compute the diversity and update Mu (Lines 9-12). The maximum size of Mu is n. The iteration terminates until it reaches a preset number of iterations.

Algorithm 1: Greedy mutant generation

Input: M: DNN model
$O = \{$GF, WS, NEB, NAI, NS$\}$: mutation operators
$R = \{0.01, 0.02, 0.03, 0.04\}$: mutation ratios
n: required number of mutants
ite: number of iterations
Output: Mu: a set of mutants

1: Initialize Mu, D, $count = 0$
2: **while** $count < ite$ **do**
3: $(o, r) = randomSelect\,(O, R)$
4: $m = mutantGenerator\,(M, o, r)$
5: **if** $|Mu|$==0 **then**
6: $Mu = \{m\}$
7: **else**
8: $D = computeCka\,(m, Mu)$
9: **if** $|Mu| < n$ **then**
10: $Mu = Mu \cup \{m\}$
11: **else**
12: $Mu = pageRank\,(D, Mu, m)$
13: **end if**
14: **end if**
15: $count++$
16: **end while**
17: **return** Mu

B. Ensemble Model Construction

Fig. 1 illustrates how we construct an ensemble. Given the training data, a model is trained with a specific archi-

tecture and parameters. By the greedy algorithm mentioned in Algorithm 1, multiple diverse mutants are obtained. In the example, we show the effect of the mutation operators GF, NEB, and NS, and the modified weights and neurons are highlighted in blue. At last, an ensemble model is built by gathering all the original models and their mutants. When performing an adversarial attack, we use the simple average strategy [12]. That is, the attack accesses each base model to obtain the gradient given an input sample, then the perturbation is calculated based on the average of all the gradients.

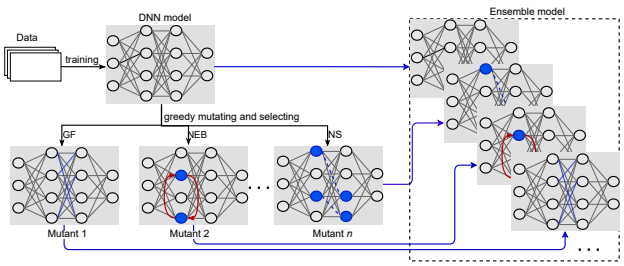


Fig. 1: Example of building an ensemble model.

IV. Experiments

A. Experimental Setup

a) DNNs and datasets: We conduct the experiments on three widely used image datasets, MNIST [13], SVHN [14], and CIFAR-10 [15]. MNIST is a 10-class grayscale handwritten digit dataset. SVHN is a real-world image dataset including 10 classes of street view house numbers. CIFAR-10 is a 10-class dataset with color images. For MNIST and SVHN, we employ the LeNet5 model as [16]. For CIFAR-10, we use VGG16 and ResNet20V1. Table I summarizes the detailed information of datasets and models. As we generate weight- and neuron-level mutants, the number of weights and neurons are also given in the third and fourth columns, respectively. For defenses, please refer to Section IV-C.

Dataset	Model	#Weights	#Neurons	#Tests	Accuracy(%)
MNIST	LeNet5	107550	236	10000	98.89
SVHN	LeNet5	136650	236	26032	88.93
CIFAR-10	ResNet20V1	270896	794	10000	90.71
	VGG16	2851008	1674	10000	91.17

TABLE I: DNNs and datasets

b) Attacks and parameter setting: Two widely used gradient-based attacks, PGD (l_∞-norm) and C&W (l_2-norm) are used for evaluation. Both attacks are implemented using the IBM ART framework [17]. Multiple perturbation levels are used for each attack to avoid the influence of parameter settings. Table II details the perturbation settings of each attack. Besides, the maximum iteration of PGD is set as 40 and 20 for MNIST/SVHN and CIFAR-10, respectively. The learning rate of C&W is 0.1, and the maximum iteration is 100 for all datasets. The default setting in ART is applied to other parameters.

Dataset	PGD ϵ	C&W c
MNIST	$0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4$	$7, 8, 9, 10, 11, 12, 13$
SVHN		
CIFAR-10	$\frac{2}{255}, \frac{4}{255}, \frac{6}{255}, \frac{8}{255}, \frac{10}{255}, \frac{12}{255}, \frac{14}{255}$	$0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4$

TABLE II: Attack configurations. "ϵ" and c stand for the maximum perturbation and initial constant, respectively.

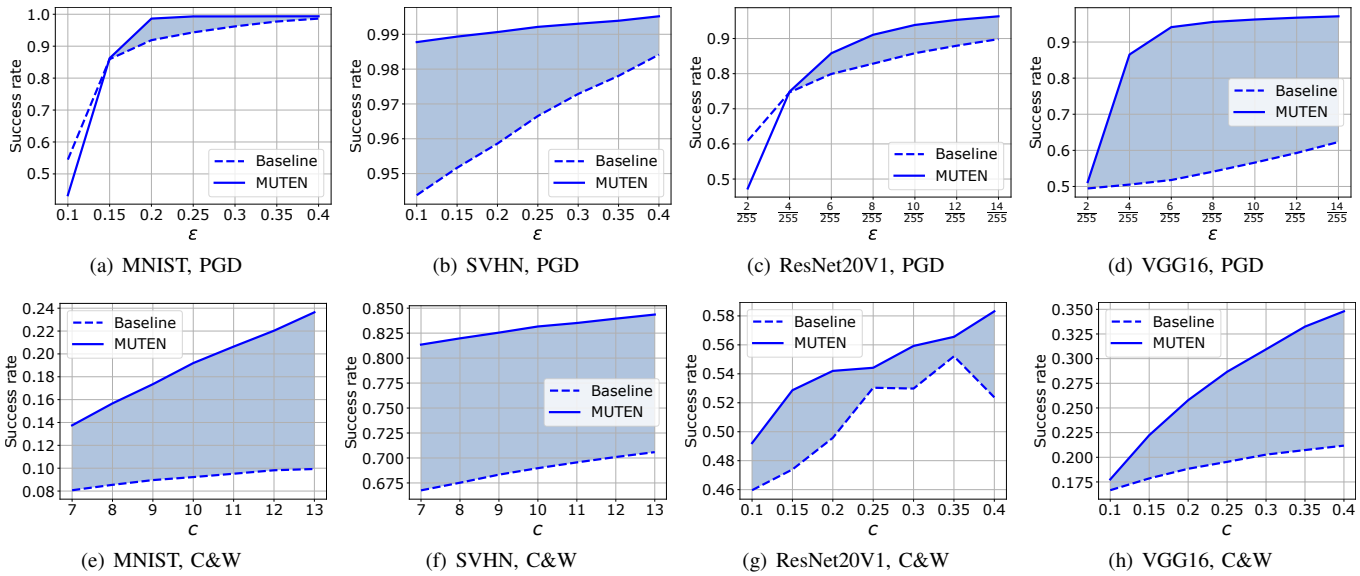
B. Performance on Clean Models

Three series of experiments are conducted to evaluate MUTEN. Each experiment was repeated five times to reduce the influence of randomness in both the creation of mutants and the application of attacks. The reported results are the average of those five runs.

a) Effectiveness: The effectiveness is measured as the success rate of applied attacks. The success rate of attacking the original model is taken as the baseline. Since the success rate converges mostly when the ensemble includes 5 mutants, we only present the result with this number. As shown in Fig. 2, overall, MUTEN achieves a higher success rate than baselines, especially as the maximum perturbation size increases. For example, in the case of PGD with VGG16, the success rate by MUTEN reaches 0.97, while the baseline increases by 0.62 with the maximum perturbation size.

b) Impact of the number of mutants: Here, we investigate how the number of mutants in the ensemble model impacts the effectiveness of MUTEN. We use the commonly used configurations ($\epsilon = \frac{8}{255}$, $c = 0.3$) for the attacks and consider a number of mutants ranging from 1 to 10. Fig. 3 shows the results. In general, the success rate of MUTEN increases as more mutant is integrated but tends to saturate quickly in the cases of PGD. Concerning the improvement of success rate, it increases from 0.32 to 0.43 in PGD, and -0.09 to 0.21 in C&W. In the case of PGD, a very high improvement of success rate can be reached with only 1 mutant, which is only the double-time of attacking the original model. Adding more mutants will increase the success rate but with a slower growth rate. In the case of the strongest attack, C&W, the success rate is lower than the baseline when the ensemble includes 1 mutant, which also happens to the other models. The reason is that for C&W, when the parameter c is small, the gradient loss has a small contribution to the loss function used by the attack algorithm. By contrast, letting the gradient loss be more important by increasing c, the success rate boosts quickly. Thus, in this case, to increase the success rate of C&W, one can either adjust c to be greater or include more mutants.

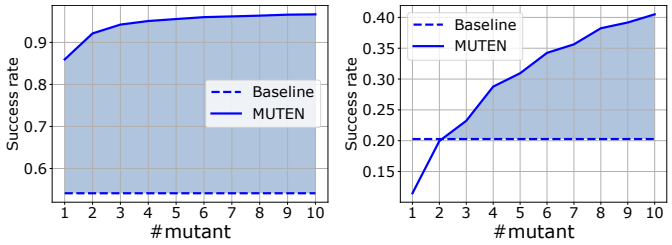
c) Diversity of mutants: Last, we investigate if the diversity contributes to MUTEN. We produce three types of mutants, diverse, random, and similar. The random mutants are generated by random selection, and similar mutants are created by limiting the test accuracy to at least 95% reserved for the original model. Fig. 4 shows the result. In general, the diverse ensemble performs the best, and the random ensemble outperforms the similar one. In the case of PGD, when the perturbation is small (e.g., $< \frac{4}{255}$), the difference between using three types of mutants is slight. By increasing the



(a) MNIST, PGD (b) SVHN, PGD (c) ResNet20V1, PGD (d) VGG16, PGD

(e) MNIST, C&W (f) SVHN, C&W (g) ResNet20V1, C&W (h) VGG16, C&W

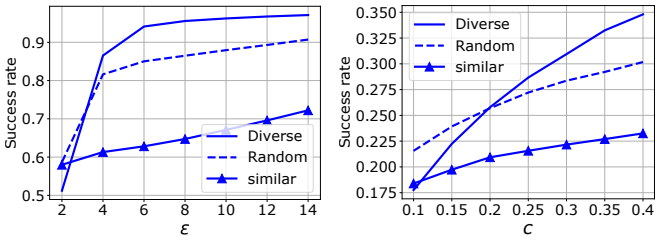
Fig. 2: Success rate VS. attack configuration. The shaded area indicates where MUTEN outperforms the baseline.



(a) PGD (b) C&W

Fig. 3: Impact of the number of mutants. Dataset: CIFAR-10. DNN: VGG16.

number of iterations of the greedy algorithm, the mutants can be more diverse, and the difference between random and diverse ensembles becomes greater.



(a) PGD (b) C&W

Fig. 4: Success rate VS. type of mutants. Each ensemble model includes 5 mutants. Dataset: CIFAR-10. DNN: VGG16.

C. Performance on Gradient Masking-Based Defenses

Additionally, we evaluate the effectiveness of MUTEN against four gradient masking-based defense approaches, Bit-depth reduction [18], JPEG compression [18], Defensive distillation [19], and Label smoothing [20]. Table III summarizes the defenses and the result of the attack success rate. The results demonstrate that MUTEN can bypass these defenses with a high attack success rate.

Defense	Accuracy	Attack	Baseline	MUTEN
Bit-depth reduction	90.98%		0.51	0.95
JPEG compression	90.54%	PGD	0.50	0.94
Defensive distillation	91.85%	$\epsilon : \frac{8}{255}$	0.61	0.96
Label smoothing	91.43%		0.67	0.82

TABLE III: Effectiveness of MUTEN against defenses. "Baseline" is DNNs without defense. Dataset: CIFAR-10. DNN: VGG16.

V. Conclusion and Future Work

We introduced a new direction of the usage of mutation testing for machine learning problems - using mutants to boost adversarial attacks. We proposed MUTEN, a novel method to build an ensemble by using diverse mutants to modify the gradient for the attacks to easier figure out the perturbation direction. The experiments on different datasets and models have demonstrated that MUTEN performs promisingly to increase the success rate of state-of-the-art gradient-based adversarial attacks with only a few mutants.

In the future, we plan to 1) evaluate MUTEN on larger datasets (e.g., ImageNet) and models (e.g., DenseNet121) to show its effectiveness, and 2) explore the direction of using mutants to build adversarial attack defense methods.

Acknowledgments

This work is supported by the Luxembourg National Research Funds (FNR) through CORE project C18/IS/12669767/STELLAR/LeTraon. Yuejun Guo is supported by the European Union's Horizon Research and Innovation Programme under Grant Agreement n° 101070303.

REFERENCES

[1] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao, and Y. Wang, "Deepmutation: mutation testing of deep learning systems," in *29th International Symposium on Software Reliability Engineering (ISSRE)*. Los Alamitos, CA, USA: IEEE Computer Society, October 2018, pp. 100–111. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ISSRE.2018.00021

[2] Q. Hu, L. Ma, X. Xie, B. Yu, Y. Liu, and J. Zhao, "Deepmutation++: a mutation testing framework for deep learning systems," in *34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, San Diego, CA, USA, December 2019, pp. 1158–1161.

[3] A. Panichella and C. C. Liem, "What are we really testing in mutation testing for machine learning? a critical reflection," ser. ICSE-NIER '21. IEEE Press, 2021, pp. 66–70. [Online]. Available: https://doi-org.proxy.bnl.lu/10.1109/ICSE-NIER52604.2021.00022

[4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *6th International Conference on Learning Representations (ICLR)*, Vancouver Convention Center, Vancouver, BC, Canada, April 2018. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb

[5] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, April 2017, pp. 506–519.

[6] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *5th International Conference on Learning Representations (ICLR)*, Toulon, France, April 2017. [Online]. Available: https://openreview.net/forum?id=Sys6GJqxl

[7] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton, "Similarity of neural network representations revisited," in *36 th International Conference on Machine Learning (ICML)*. Long Beach, California: PMLR, 2019. [Online]. Available: https://proceedings.mlr.press/v97/kornblith19a/kornblith19a.pdf

[8] C. B. Moler, *Experiments with MATLAB*. Society for Industrial and Applied Mathematics, 2011.

[9] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2017. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP.2017.49

[10] J. Wang, G. Dong, J. Sun, X. Wang, and P. Zhang, "Adversarial sample detection for deep neural network through model mutation testing," in *ICSE*, ser. ICSE '19. IEEE Press, May, p. 1245–1256. [Online]. Available: https://doi-org.proxy.bnl.lu/10.1109/ICSE.2019.00126

[11] J. Chen, Z. Wu, Z. Wang, H. You, L. Zhang, and M. Yan, "Practical accuracy estimation for efficient deep neural network testing," *ACM Transactions on Software Engineering and Methodology*, vol. 29, no. 4, October 2020. [Online]. Available: https://doi-org.proxy.bnl.lu/10.1145/3394112

[12] N. Demir, "Ensemble methods: elegant techniques to produce improved machine learning results," https://www.kdnuggets.com/2016/02/ensemble-methods-techniques-produce-improved-machine-learning.html/2, 2016.

[13] 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, November 1998.

[14] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

[15] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Tech. Rep., 2009.

[16] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples," in *ICML*, vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, July 2018, pp. 274–283.

[17] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, and B. Edwards, "Adversarial robustness toolbox v1.2.0," *CoRR*, vol. 1807.01069, 2018.

[18] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," in *6th International Conference on Learning Representations (ICLR)*, Vancouver Convention Center, Vancouver, BC, Canada, April 2018. [Online]. Available: https://openreview.net/pdf?id=SyJ7ClWCb

[19] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2016, pp. 582–597. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP.2016.41

[20] D. Warde-Farley, "1 adversarial perturbations of deep neural networks," 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:28912221